Optimising Seismic Imaging via Bilevel Learning: Theory and Algorithms

Downing, Shaunagh

Award date:
2022

Awarding institution:
University of Bath

Optimising Seismic Imaging via Bilevel Learning: Theory and Algorithms

submitted by

Shaunagh Downing

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Mathematical Sciences

January 2022

COPYRIGHT

Declaration of any previous submission of the work

The material presented here for examination for the award of a higher degree by research has not been incorporated into a submission for another degree.

Signature of Author .....

Shaunagh Downing

Declaration of authorship

I am the author of this thesis, and the work described therein was carried out by myself personally, under the supervision of Silvia Gazzola, Ivan Graham, and Euan Spence.

Signature of Author .....

Shaunagh Downing

Summary

Seismic inversion is the inverse problem of determining properties of the Earth's subsurface from measurements of waves propagating through it. A standard algorithm for solving this inverse problem is called full waveform inversion (FWI). FWI computes a model describing the subsurface by minimising the misfit between actual measurements and numerically-predicted data plus one or more regularisation terms, which are added to deal with the ill-posedness of the inverse problem. The implementation of FWI requires the a priori choice of a number of parameters, including the positions of sensors for the wave measurement and the regularisation parameters. A problem of great practical interest, which is not considered in the standard approach to FWI, is the optimal positioning of the sensors in order to obtain the best outcome from the seismic imaging process. In this thesis, it is shown that, given a set of training models of realistic wave velocities, one can learn the optimal sensor positions and regularisation parameters, thus optimising the performance of the standard FWI reconstruction algorithm. We establish a novel fundamental theory underpinning the solution to this sensor optimisation problem by placing it in the framework of bilevel learning. In our formulation, the upper-level objective function measures the misfit in the reconstruction of the training models via FWI, so that FWI itself constitutes the lower-level optimisation problem. We propose to solve the bilevel problem with a gradient-based optimisation method. Our chosen forward problem is the acoustic wave equation, which we solve in the frequency domain (via the Helmholtz equation).

This thesis contains contributions both in the theory and application of this bilevel learning problem. In particular, for the theory, this thesis contains the following novel contributions:

- We give sufficient conditions, in terms of the regularisation parameters in the lower-level/FWI problem, for the lower-level problem to have a unique solution.
- We derive a formula for the gradient of the upper-level objective function and show that this requires solving systems involving the Hessian of the lower-level problem, for which ill-conditioning is mitigated by the choice of lower-level regularisation.
- We prove smoothness properties of the upper-level objective function by exploiting the theoretical properties of the partial differential equations modelling the propagation of acoustic waves in the frequency domain.
- We show that, under assumptions on the symmetry of the domain, model and source positions, the optimal set of sensor positions is symmetric.

Our main novel contributions to the application aspect of this bilevel problem are the following:

- We design a bilevel learning algorithm for optimising sensor positions and the Tikhonov regularisation parameter in FWI.

- We give a complexity analysis for the bilevel algorithm, involving a study of the number of forward solves needed by the algorithm.
- We propose a bilevel frequency continuation strategy to improve the performance of the bilevel algorithm.
- We propose a preconditioning strategy for the systems involving the Hessian which have to be solved at each step of the upper-level gradient descent.
- We implement the bilevel algorithm and provide illustrations of the algorithm on test problems.

Acknowledgements

There are many people who have supported me along my PhD journey, and I am deeply grateful to all of you. Go raibh míle maith agaibh go léir.

Firstly, I would like to thank my supervisors, Euan Spence, Ivan Graham and Silvia Gazzola, for all their support and guidance throughout my PhD. Thank you for always making time for me, for your assistance and advice at every stage of my project, and for helping me to become a better mathematician and researcher. The help you have provided me over the past few years has been invaluable. I am very lucky to have had such a great team of supervisors.

I am thankful to EPSRC for funding my studentship through SAMBa, and to Evren Yarman for originally proposing the research question that I based my thesis on.

Along with Evren, I am also grateful to James Rickett and Kemal Ozdemir for sharing so much of their expert knowledge with me and for all their insightful conversations about the applications of my work.

There are many people at the University of Bath that I wish to thank. I would like to thank the SAMBa team, for giving me the opportunity to be a part of this Centre for Doctoral Training and for their support throughout. During my research, I made extensive use of the High Performance Computing (HPC) facilities at Bath and I am grateful for the support provided by the HPC team. I would also like to thank Matthias Ehrhardt and Tony Shardlow for providing me with useful comments and suggestions on my work, which I appreciate very much.

On a personal level, I would like to express my gratitude to my amazing family. To my parents, James and Patricia, thank you for your unwavering love and support, for always believing in me, and for encouraging me to follow my passions. Thank you for every sacrifice you have made for me and every opportunity you have given me in life. To my siblings, Caoimhe and Eoin, thank you for being such great friends to me and for being supportive of everything I do. And to my uncle Noel, thank you for helping me get back home to Kenmare whenever I could!

Finally, I would also like to thank my partner Matt, who has been such a wonderful support throughout my PhD. Thank you for being by my side during all the ups and downs of my PhD journey and for everything you have done for me along the way.

I want to dedicate this thesis to the memory of those close to me that I lost over the last few years - my grandmother Rita Downing, and my uncles Jeremiah Downing and Donal Shea.

Contents

Chapter 1

Introduction

1.1 Structure and Overview of Thesis

The thesis is arranged as follows. **Chapter 1** gives a general introduction to seismology and seismic imaging, which leads to the motivation for optimal parameter choice for the seismic imaging procedure in Section 1.5.

Chapter 2 is a combination of an overview of existing and novel theory for the seismic imaging procedure Full Waveform Inversion (FWI). Section 2.2 contains the definition and formulation of FWI. Sections 2.3 and 2.4 present gradient and Hessian derivations respectively, which may be used in optimisation methods as part of FWI. Sections 2.4.3 and 2.4.4 contain discussion and novel results about the structure of the Hessian and how regularisation can be used to make the Hessian positive definite, and hence the FWI problem convex; these results become important in Chapter 3. We focus considerably on the FWI Hessian in Chapter 2 as this matrix also appears in novel formulae derived in Chapter 3. In Section 2.5, the full FWI algorithm is outlined and the implementation technique of frequency continuation is discussed. We then present some applications of the FWI algorithm in Section 2.6.

Chapter 3 presents the parameter optimisation problem for FWI. We begin with a review of seismic survey design and current approaches to optimal sensor placement in Section 3.1, before outlining our new idea for choosing optimal sensor positions, and incorporating the choice of an optimal regularisation parameter into the process. This chapter is concerned with the formulation of this parameter optimisation problem, the derivation of relevant formulae, and the analysis of the problem. All work in Chapter 3 is novel. This original work includes the formulation of the sensor placement and regularisation parameter optimisation problem in the framework of bilevel learning, where the optimal parameters are learned from a training set. In this bilevel learning framework, one level (the lower-level) is the FWI problem, and the other level (the upper-level) is the parameter optimisation problem. We propose a gradient-based local optimisation method for solving the bilevel problem, derive a novel formula for the gradient of the upper-level objective function and apply the adjoint-state method to yield an efficient algorithm for its computation. We include an analysis of the cost of computing the gradient in terms of the number of PDE solves required. We provide

analysis of the smoothness of the bilevel problem with respect to the optimisation variables, showing that the bilevel problem can indeed be solved using a gradient based optimisation method. The formulation and analysis described is first presented for the optimisation of sensor placement in Sections 3.3 and 3.4, before the optimisation of the regularisation parameter is included in Section 3.5. Some experiments are presented in Section 3.6, demonstrating that the bilevel formulation and gradient-based local optimisation method works well to improve the FWI reconstruction by optimising sensor positions, and works even better when the optimisation of the regularisation parameter is included in the process.

Chapter 4 is an original analysis of the symmetry properties of the bilevel problem. We show that under certain assumptions, the solutions to both the lower- and upper-levels have symmetric properties. We demonstrate the results in this chapter numerically and show how to exploit symmetry to make solving the bilevel problem more efficient.

In **Chapter 5**, we present the algorithms used to solve the bilevel problem and the implementation details of these algorithms. This includes techniques for improving the efficiency of the implementation, many of which are original. We have developed a novel bilevel frequency continuation approach that is shown to improve the performance of the bilevel algorithm by avoiding local minima on the upper and lower-levels. The development of this approach and examples of how the technique works are contained in Section 5.1. In Section 5.2, we discuss the computation of the upper-level gradient, specifically how to solve the linear system involving the FWI Hessian that arises in the gradient formula. We propose two novel preconditioning strategies to reduce the number of iterations taken to solve this system. We demonstrate that both preconditioning strategies work effectively to speed up the upper-level gradient computation, and hence to speed up the overall bilevel algorithm. Section 5.3 is concerned with the parallelisation and scaling of the bilevel algorithm. We present measurements of the runtime of the bilevel algorithm to show that the algorithm scales well in parallel. We also provide a breakdown of the computational time spent on different parts of the algorithm. Section 5.4 presents the full bilevel algorithm that we have developed, and any further important implementation details.

Chapter 6 contains two larger-scale parameter optimisation experiments that employ the theory and implementation described throughout the rest of the thesis. One experiment involves applying our bilevel algorithm to a training set of various models with some shared characteristics. We show that we need relatively few training models to produce huge improvements in the FWI images. We include an analysis of how well the optimal parameters work on models outside of the training set, as we test on cases that are further and further from the training models. Our second experiment involves applying our algorithm to a geophysical problem based on the Marmousi model. Although this problem is more difficult, we still find parameters that produce an improvement in the FWI reconstructions of the training and testing sets.

1.2 Introduction to Seismology

Seismology is the scientific study of mechanical vibrations in the Earth. Traditionally, seismology has been concerned with the measurement, monitoring and prediction of earthquakes, with modern seismology starting with the study of the Lisbon earthquake in 1755. The field has been in development ever since, with theoretical and practical developments, as well as growth in its range of applications. A review of these developments is given in this section, based on [8, Chapter 1] and [23].

Theoretical advancements in seismology involved the mathematical study of seismic waves. The key discoveries are summarised here. In the 1800's the theoretical foundations were laid for the mathematical description of elasticity and wave propagation in elastic solids. The full theory of 3-dimensional, stressed, elastic objects was developed by Claude-Louis Navier and Augustin-Louis Cauchy in 1821-22. In 1828, Poisson theoretically showed the existence of both longitudinal and transverse waves (also termed P and S waves) in elastic solids. In 1885, Rayleigh predicted the existence of a new type of wave, which exists on the boundary, or surface, of elastic materials, which he suggested might play an important part in earthquakes. These waves became known as Rayleigh waves. In 1911, A.E.H. Love predicted the existence of another type of boundary wave, later becoming known as Love waves.

In addition to these theoretical mathematical developments, large practical advancements had been made. Robert Mallet, an Irish engineer, was considered to have laid the foundation of instrumental seismology. He coined the term 'seismology', published the first map of world earthquake occurrence, made the first systematic attempt to apply physical principles to the movements of seismic waves, and carried out a number of experiments to determine the speed of seismic propagation in different soils. More details on Mallet's work are conatined in [76] and [140].

The greatest practical advancement in quantitative seismology came with the development of the seismograph - a device used to measure and record vibrations in the Earth. The earliest seismograph was built in 1841 by J. Forbes, but these devices were primitive. More sophisticated seismographs were constructed in the 1880's by J.Ewing, J. Milne and T. Grey for the measurement of earthquakes in Japan. More sophisticated seismographs continued to be developed over the following years. During an earthquake, a rupture in the Earth generates seismic waves that travel outward. Seismographs on the Earth's surface could now measure the amplitude and arrival times of these waves, and through combining measurements at multiple locations, earthquake epicentres could be located. By 1889, seismographs were sensitive enough to record earthquake vibrations on the other side of the world, demonstrated by E. von Rebeur-Paschwitz who measured earthquakes in Japan with a seismograph in Germany. In the 1900's, seismographs began being used for applications other than earthquakes. In 1921, J.C. Karcher was the first to conduct a seismic imaging experiment. The idea was to use a man-made explosion (dynamite) as a source of seismic waves and record the waves with seismographs to conduct a survey of underground structures. This was seen as the beginning of exploration seismology. To this day, this seismic method remains the most popular method for characterising the subsurface.

Since the 1960's, the field of seismology has advanced quickly due to the combined improvements in instrumentation, computing power, and the mathematical theory of seismic waves. Quantitative seismology today involves many highly advanced techniques, which are continually being developed, including high-quality data collection, detailed models of wave propagation, inverse problem theory and modern high performance computing.

Nowadays, seismology is used extensively in mineral prospecting and exploration for oil and natural gas. It is also commonly used to help detect groundwater, in civil engineering to aid in the design of earthquake-resistant buildings, and to asses the integrity of foundation structures [102]. It has also had several other interesting applications, from locating heavy artillery positions of the enemy during World War I [18] to landmine detection [184]. The concepts used in seismology have also been applied in other fields, for example medical imaging (see [81] and [120]).

For a more detailed introduction into the concept and history of seismology, see [8], [23] and [112].

1.3 Underlying Principles of Seismic Imaging

The overall aim of seismic imaging is to find a structural image of the interior of a body. Exploration seismology is an important and widespread application of seismic imaging. The goal in exploration seismology is to image structures in the subsurface and determine the values of material parameters of these structures, in the search for mineral deposits (such as oil, gas, water and geothermal reservoirs) and archaeological sites, or to acquire geological information for engineering applications [161]. Seismic exploration, and more generally the study of the Earth, through seismic imaging is based on the propagation of waves. Waves are influenced by the medium in which they propagate so that the analysis of their propagation reveals information on the zone in which they travelled. The physical phenomena that make seismic imaging possible are the reflection and refraction of waves. When a wave front reaches an obstacle or a discontinuity/boundary interface in the subsurface, a part of the wave is reflected, and part of the wave is refracted, either transmitting across the interface into the next medium or propagating along the interface. This behaviour is what makes it possible to emit waves from a location on the surface, and receive information back, allowing seismic experiments to be set up such that the characteristics of the subsurface can be inferred based on measurements made at the surface.

Seismic acquisition refers to generation and recording of seismic data. For exploration purposes, acquisition involves a configuration of sources and sensors. The *source*, often positioned at the surface but sometimes positioned in a well, artificially generates waves which are directed into the ground. The seismic waves then propagate through the Earth, and when they reach subsurface boundaries and changes in rock mechanics, parts of the wave are reflected and refracted (as described above). The reflected and refracted waves are detected by *sensors*, usually positioned at the surface or occasionally in a well below the surface. The sensors can record the properties of the returning wave, such as its strength, and the time it has taken to travel from the source, through the layers of

rock in the Earth's crust, and back to the surface. The reflections/refractions that come from transitions between media in the subsurface are governed by differences in their properties, such as density, velocity and elasticity. Therefore, the measurements made by sensors are interpreted to reveal information about these different material properties and, after some processing, can be transformed into images of the subsurface beneath the seismic survey. Although there are various methods to recover the investigated parameters, the goal is the same: the discovery of the unknown underground.

In summary, although we cannot see beneath the ground, we can take advantage of wave propagation during seismic surveys to get an image of the subsurface and determine properties of the rock layers indirectly.

1.4 Seismic Methodology

Now that we have described the main principles of seismic imaging, we take a closer look at seismic acquisition in practice. Seismic data is acquired during a procedure called a seismic survey. Seismic acquisition can take place either on the surface of the Earth (land acquisition), as shown in Figure 1.4.1, or offshore (marine acquisition), as shown in Figure 1.4.2. In this section, we focus on marine acquisition.

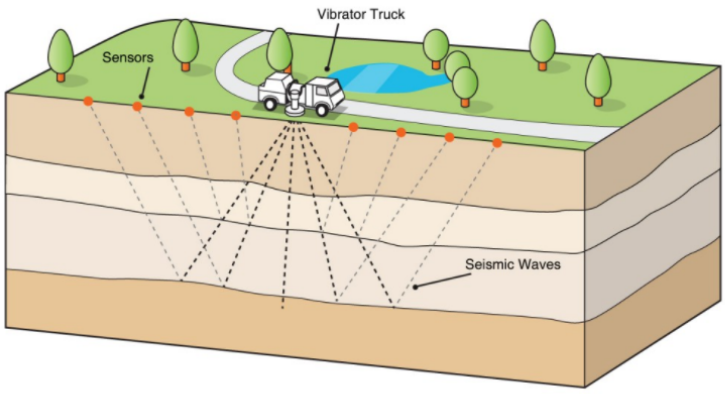


Figure 1.4.1: *Illustration of a land, or onshore, seismic acquisition [2]. A vibrator truck generates acoustic waves which are directed into the Earth. These waves reflect off the various ground layers and are recorded by the network of sensors on the surface.*

Marine seismic acquisition is carried out using large ships called seismic survey vessels [96]. These vessels are built with special features that aid in the monitoring and processing of seismic waves, and sail along predetermined paths during surveys. The vessel tows seismic cables, or streamers, behind it, which can be several kilometres long and on which the sensors are mounted.

Most marine surveys use air-guns as a source. The air-gun creates a seismic wave by discharging air under very high-pressure into the water. The total energy of the source is specific to each survey. The stronger the source, the deeper the structures

that can be imaged. The sources are therefore chosen so as to illuminate the subsurface sufficiently while minimising environmental disturbance.

The sensors are towed behind the ship and are used to detect the reflected seismic energy. The typical sensor used in marine surveys is the hydrophone. These devices have a piezoelectric element that converts changes in water pressure into an electrical signal. The signal is digitised and transmitted to the recording system in the vessel. Geophones can also be used as sensors. These types of sensors are sensitive to local particle displacements and record displacement amplitude as the wave propagates through the medium. Geophones are more common for land surveys but are sometimes used alongside hydrophones in marine surveys [74].

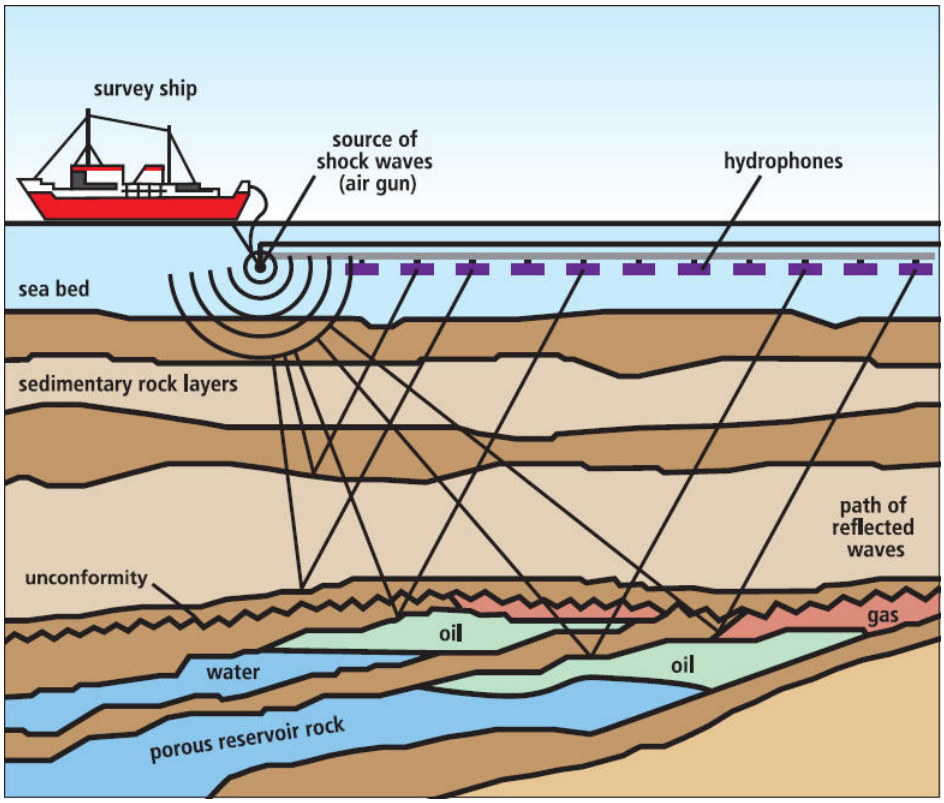


Figure 1.4.2: *Schematic of marine, or offshore, seismic acquisition [1].*

Seismic surveys can vary greatly in their method. However, there are two main types of operation; 2D and 3D [96].

2D Seismic Operation: This is the simpler and less expensive method, in both processing and acquisition, compared to the 3D operation. In this method, a single seismic cable, or streamer, and a single source are towed behind the survey vessel, and the data is acquired along a line of sensors. The reflections from the subsurface are assumed to occur directly below the sail line (the line traversed by the ship), providing an image in two dimensions, hence the name '2D'. The 2D method is useful in obtaining a general understanding of the subsurface structure, however it does not always produce an accurate subsurface image.

3D Seismic Operation: A 3D survey covers a specific area that has been chosen with the help of the preliminary 2D survey data. The 3D surveys are carefully planned to ensure the survey area is accurately defined, with known geological targets determined from previous data. This planning generates a map of survey boundaries and direction of sail lines.

The 3D operation is equivalent to acquisition from several 2D lines running in parallel close together. The survey vessel tows several sources and parallel streamers, separated by up to 50 metres, as shown in Figure 1.4.3. Therefore, 3D acquisition is achieved by a single sail line. Generally, groups of sail lines are traversed in a survey, with a typical separation of the order of 200 - 400 metres [96]. 3D surveys generate significantly more data than the 2D case. Powerful computers are necessary for the processing of this large volume of data into a 3D map of the subsurface. The detailed information about the subsurface provided by the 3D seismic operation makes it the preferred method of seismic survey, accounting for 95% of all marine seismic survey data worldwide [96].

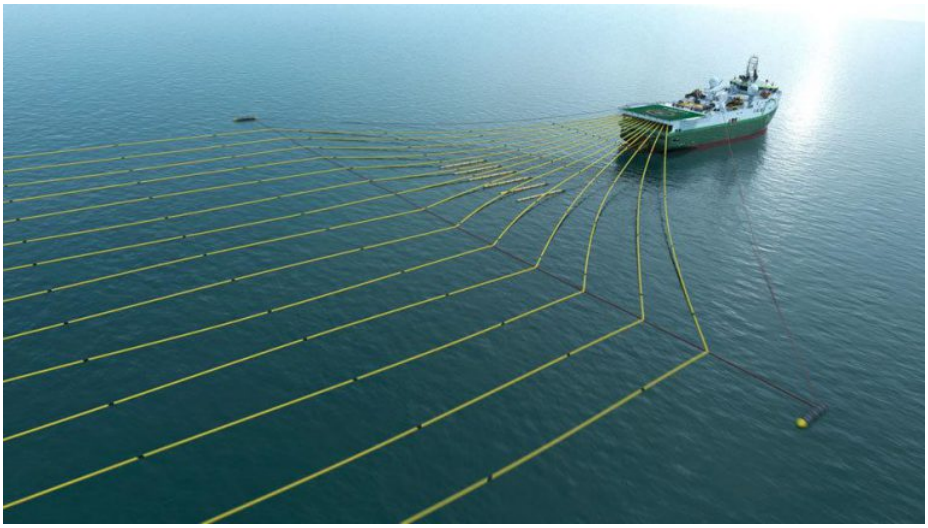


Figure 1.4.3: *3D marine seismic acquisition [3].*

One of the most common uses of seismic survyes is in the search for hydrocarbon resources, and most commercial seismic surveys are carried out by the oil and gas industry [96]. Seismic surveys serve multiple purposes in this sector. Initially, the data is used to identify subsurface structures that are likely to contain hydrocarbons, leading to new drilling locations. In areas of existing production, the survey can be used to find finer-scale details about the subsurface, for example, to establish the areas of the reservoir not drained by existing wells, to estimate reserves, and to monitor the movement of reservoir fluids in response to production [133].

1.5 Thesis Motivation

It is vital for oil companies to plan seismic surveys carefully, to ensure survey objectives are achieved at the lowest possible cost. Poor planning of seismic surveys has been one of the main factors resulting in an estimated 10% of surveys failing to achieve their primary objective [13]. Therefore, careful planning of surveys is essential in achieving cost-effective acquisition and processing, as well as high quality data. Survey designers must establish the best way to image the subsurface, considering, for example, locations and types of sources and sensors, time required for acquisition and environmental issues.

This thesis considers the problem of optimising the outcome of seismic acquisition through optimal sensor placement and optimal parameter choice in the seismic imaging algorithm. This is a topic of practical interest in petroleum prospecting, and is not currently considered in the standard mathematical approach to seismic imaging. Determining optimal sensor locations can inform several aspects of designing seismic surveys, for example, the acquisition trajectory of the boat, cable spread behind the boat and sensor locations on individual cables. Sensor location optimisation will have the benefits of optimal utilisation of the acquisition system, while at the same time improving data quality. Optimal parameter choice in the seismic imaging algorithm will ensure that the data collected during seismic acquisition will produce an image of the subsurface that is as accurate as possible.

The key idea in our approach to parameter optimisation is to exploit *prior information* about the subsurface, in the form of training images, to create the optimal setup for seismic acquisition. Prior information may be available due to 2D surveys, previous drilling or exploratory wells, for example. In fact, many exploration projects take place in areas known as mature fields [4], where a great deal of information is available about the subsurface. Mature fields are areas where exploration has been ongoing for years, and which have already been drilled for hydrocarbons. Areas within these fields are often prioritised for a closer geophysical exploration, based on the knowledge geologists already have on the area [133], and to control the risk of the exploration project failing [4]. In our parameter optimisation strategy, we assume we have access to prior information about the subsurface and that this information is accurate. We label this as *training* data or training images. We use this training data to *learn* 'good' parameters, where 'good' parameters recover subsurface information that sufficiently matches the known information (i.e., matches the training data).

Chapter 2

Full Waveform Inversion

Chapter Summary: This chapter provides an overview of Full Waveform Inversion (FWI). We start with a general introduction to inverse problems (§2.1), before focusing on some key aspects of FWI. We discuss the main features of FWI in (§2.2), including the forward modelling step, where we include a specific forward problem example (the Helmholtz equation) and a brief discussion of its discretisation, the formulation of the objective function, and the optimisation step. We include a derivation of formulae for the gradient (§2.3) and Hessian (§2.4) of the FWI objective function, and formulae for Hessian-vector products (§2.4.2). We provide a detailed discussion of the properties of the Hessian in §2.4.3, including both theory and numerical experiments. In §2.4.4 we provide novel results on upper and lower bounds for the eigenvalues of the Hessian, and hence derive a condition on a regularisation parameter for ensuring its positive-definitiveness. We write the FWI algorithm in §2.5, and include some examples of FWI reconstructions in §2.6.

2.1 Introduction to Inverse Problems

2.1.1 Definition

A forward problem involves taking inputs and, under some known process, producing corresponding outputs, which can be measured. An inverse problem takes the measurements from this known process, and infers what inputs produced them. In other words, a *forward* problem is one in which we find an effect from a known cause and an *inverse* problem is one in which we try to determine the cause of an observed effect.

Mathematically, an inverse problem can be formulated as follows. Let f denote the process that takes a set of inputs x and produces a corresponding set of outputs y. f can be some mathematical function, or a mathematical model of some physical phenomena. The relationship between the inputs x and outputs y can be written in the form $f(x) = y$. The forward and inverse problems can then be expressed mathematically as:

- **Forward Problem**: Given inputs x and the process f, find the outputs y.

- **Inverse Problem**: Given the outputs y and the process f, find the inputs x.

Note that the application of f to the inputs in the forward problem can involve the solution of a differential equation to find the output y. When no exact solution is available, the forward problem is solved numerically to approximate y to a certain degree of accuracy. Therefore, f may represent an approximation. The inverse problem is generally much more difficult to solve than the forward problem, due to the *ill-posedness* of the problem (see the next subsection).

A more in-depth look at inverse problems can be found in [177] and [168].

2.1.2 Ill-posedness

Hadamard introduced the concept of a *well-posed* problem in [86]. For the inverse problem defined as in Section 2.1.1, i.e., as
'*Given known data y, and a process/governing law f, find the solution x that satisfies $f(x) = y$*',
the problem is well-posed when the following conditions are satisfied:

- *Existence:* There exists a solution of the problem (for all y, there exists x such that $f(x) = y$);
- *Uniqueness:* The solution is unique (if $f(x) = f(w) = y$, then $x = w$)
- *Stability:* The solution of the problem, x, depends continuously on y, and is insensitive to small changes in y.

A problem violating any of these conditions is called *ill-posed*. It turns out that the most interesting and important inverse problems are ill-posed. It is still possible to obtain meaningful and accurate solutions from ill-posed problems, using techniques such as regularisation and iterative methods. A more detailed explanation of inverse problems and their ill-posedness can be found in the survey paper [99].

2.1.3 Inverse Problems in Seismic Imaging

We now focus on inverse problems in the context of seismic imaging (see [157] and [126]). The goal of seismic inversion is to transform measured data into knowledge about the physical world. In other words, we are trying to recover some information about the make up of the Earth's crust, or subsurface, that we cannot directly access. In this case, the *data* $\mathbf{d}$ are measurements of the seismic wavefield (e.g displacement or pressure) and we aim to transform these into a *model* $\mathbf{m}$. The model is a set of variables that describe the properties of the subsurface. This model can encode any physical property that controls the behaviour of the wavefield, such as elastic properties, density or velocity/wavespeed of the subsurface.

For such a transformation to be possible and reliable we need a physical law linking the model parameters $\boldsymbol{m}$ and the observed data $\mathbf{d}$. The forward problem can be a wave-equation, which provides the mathematical connection between the model parameters

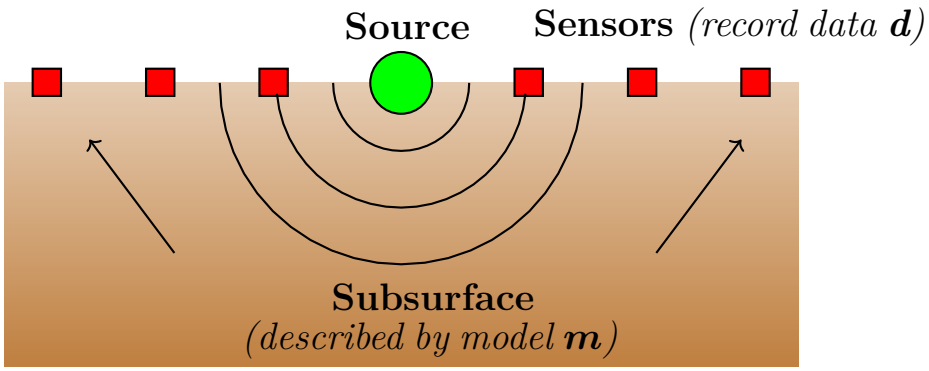


Figure 2.1.1: *Illustration of seismic acquisition for a two-dimensional domain with a single source positioned at the surface (represented by the green circle). The wave (indicated by semi-circles) propagates from the source through the subsurface area of interest, where structures in the Earth reflect part of the wave (indicated with arrows). The sensors (represented by the red squares) record the resulting signal, providing the data* $\boldsymbol{d}$*. The subsurface is characterised by the model* $\boldsymbol{m}$*, which controls the behaviour of the wavefield.*

and the seismic wavefield. We can express the *modelled data*, or predicted data, coming from the forward model as

$$\mathbf{d}^{mod} = \mathcal{F}(\boldsymbol{m}). \tag{2.1.1}$$

The operator $\mathcal{F}$ is the forward modelling operator, and can be thought of as performing two steps - solving the wave equation for specific model parameters $\boldsymbol{m}$, and extracting the modelled wavefield at observation points (i.e., sensor locations). For wave propagation, the forward problem is typically deterministic, which means that, given the model parameters and assumptions we make on the physics of the wave equation (e.g., acoustic or elastic), $\mathbf{d}^{mod}$ is uniquely defined. The same cannot typically be said for the inverse problem, i.e., the transformation from the data to the model.

Due to noise in the observed data, a limited number of measurements compared to the size of the subsurface, or simplifications in the physics of the wave equation, it is often not possible for $\mathcal{F}$ to completely explain the observed data, and so $\mathbf{d}$ will not equal $\mathbf{d}^{mod}$ exactly. Therefore, one aims to find the model $\boldsymbol{m}$ that best describes the observed data $\mathbf{d}$, i.e., to find the model $\boldsymbol{m}$, such that $\mathbf{d} \approx \mathcal{F}(\boldsymbol{m})$. Optimisation methods can be applied to solve the inverse problem by finding a model that minimises an appropriate function (that involves the difference between $\mathbf{d}$ and $\mathcal{F}(\boldsymbol{m})$).

The method of Full Waveform Inversion (FWI) is often employed to solve seismic imaging inverse problems. FWI combines the forward modelling and optimisation steps described above. The following section provides a more detailed review of the different aspects of FWI.

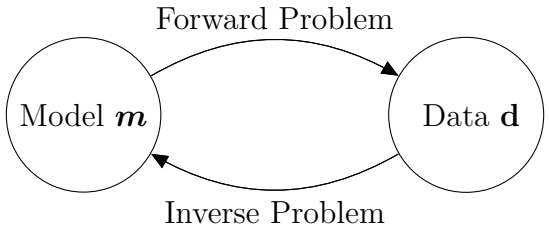


Figure 2.1.2: *The transformation from the model to the data is found by solving the often deterministic forward problem. Obtaining the model from the data is referred to as the inverse problem.*

2.2 Full Waveform Inversion Overview

2.2.1 FWI Definition

FWI is a non-linear, ill-posed, data-fitting procedure, where the aim is to *iteratively minimise* the difference between the *observed seismic data*, acquired in a seismic survey, and the *modelled seismic data* (also called predicted or computed data), generated from the numerical modelling of waves. The result is a geological model, which provides quantitative information about the make-up of the subsurface and a structural image of the subsurface.

The FWI framework is comprised of two main parts - the **forward problem** and the **inverse problem**. The forward problem involves the generation of the modelled data through numerically solving a wave equation. The inverse problem involves using an iterative optimisation method to find the model that minimises the misfit between forward model predictions and measured data, defined in some norm.

The FWI solution is the *optimal model*, i.e., it is the model that reproduces the observed data most closely. The optimal model is therefore assumed to be the model that best describes the real world.

The following sections provide a mathematical overview of each aspect of FWI - the forward modelling (including discretisation), the definition of the objective function to be minimised, and the solution of the FWI problem. Some background on the development of FWI as well as some current research is included in Appendix A.

2.2.2 Forward Modelling

Seismic imaging relies on waves propagating in the subsurface, where the movements of the waves are determined by the properties of the medium. The modelling of seismic waves is therefore essential in imaging the structures that the waves propagate through. The forward modelling step involves solving the forward problem. In this section we will review the forward problem, which mathematically describes seismic waves that are generated by a source, and travel through the Earth.

Seismic waves may be described by a partial differential equation (PDE) called a *wave equation*. The formulation of this PDE occurs naturally in the time domain,

however, in this thesis we choose to formulate the problem in the frequency domain. The PDE involves a wave operator, which depends on the type of medium, for example acoustic or elastic. We can represent a frequency domain wave equation generally, at selected frequency ω with the operator $\mathcal{A}_\omega$, so that the wave equation may be written as follows,

$$\mathcal{A}_\omega(m)u = f, \tag{2.2.1}$$

where m is the geological/subsurface model, f represents a source, and u is the *wavefield*. The wavefield is a term which represents the aspect of the wave we are modelling, and so may be the displacement (elastic wave equation) or pressure (acoustic wave equation) field. Note that spatial dependency is not written explicitly in (2.2.1), but the model $m = m(x)$ varies in space due to the inhomogeneity of the subsurface. The type of wave equation and definition of the model are determined by the definition of the wave operator $\mathcal{A}_\omega(m)$. The notation in (2.2.1) is kept general to demonstrate that results and analysis presented later in the thesis for the Helmholtz equation, can also be applied to other frequency domain wave equations. The FWI steps are identical for all types of medium.

Remark 2.2.1. ***Helmholtz Equation:*** *In this thesis we focus specifically on the Helmholtz equation. This is the acoustic frequency domain wave equation, with constant density. The underlying concept for acoustic FWI is that acoustic waves propagate at different velocities through different materials. Therefore, the model for this forward problem is related to the wavespeed. The model is often termed the squared slowness, and is defined as*

$$m(x) = \frac{1}{c^2(x)} \tag{2.2.2}$$

The wave operator for Helmholtz equation is

$$\mathcal{A}_\omega(m) := -(\Delta + \omega^2 m). \tag{2.2.3}$$

We note that the Helmholtz equation is widely studied and applied as the forward equation in the FWI problem, for example see [70], [17] and [55].

We include some more details on seismic waves and their mathematical formulation in the appendices. A brief description of the physics of seismic waves is included in Appendix B, the formulation of the elastic and acoustic wave equations and the relationship between them is contained in Appendix C and the frequency domain formulation of the wave equations are discussed in Appendix D.

2.2.2.1 Forward Problem of Interest

Although results and formulas presented in this thesis can apply to any formulation of the wave equation, with any appropriate source and boundary conditions, the forward problem that this thesis focuses on is the Helmholtz problem (acoustic medium) with

impedance boundary conditions. Impedance boundary conditions are an approximation to the Sommerfeld radiation condition (see Appendix D, Equations (D-7) and (D-8)), used when artificially truncating an unbounded domain.

We present the notation and setting of this specific forward problem here, along with definitions and propositions required in future theorems.

Notation	**Meaning**	**Details/Properties**
Ω	Domain of interest	$\Omega \subset \mathbb{R}^d$, with $d = 2$ or 3
$\partial\Omega$	Domain boundary	Lipschitz boundary
$\mathcal{S}$	Set of source positions	$\mathcal{S} \subset \Omega$, $N_s = \#\mathcal{S}$
s	Specific source position	$s \in \mathcal{S}$
$\mathcal{P}$	Set of sensor positions	$\mathcal{P} \subset \Omega$, $N_r = \#\mathcal{P}$
p	Specific sensor	$p \in \mathcal{P}$ Sometimes we enumerate the sensors as p_j, $j = 1, \ldots, N_r$.
$\mathcal{W}$	Finite set of angular frequencies	$\mathcal{W} \subset \mathbb{R}$, $N_\omega = \#\mathcal{W}$
ω	Specific angular frequency	$\omega \in \mathcal{W}$
m	Model to be reconstructed	Depends on M real parameters m_k as: $m(x) = \sum_{k=1}^{M} m_k \beta_k(x)$, $x \in \Omega$
$\boldsymbol{m}$	Vector of model parameters m_k	$\boldsymbol{m} \in \mathbb{R}^M$
β_k	Non-negative real-valued local basis functions defined on Ω	$\sum_k \beta_k(x) = 1$ for all $x \in \overline{\Omega}$

Table 2.2.1: *Notation used for the forward problem*

The PDE problem of interest is posed in continuous form without discretization, but the model is assumed to depend on several parameters. We restate the definition of the model from Table 2.2.1 to emphasise this,

$$m(x) = \sum_{k=1}^{M} m_k \beta_k(x), \qquad \sum_k \beta_k(x) = 1. \tag{2.2.4}$$

A similar representation of the FWI model as a sum of basis functions, to provide a translation between continuous and discrete model space, can be found in [71].

Remark 2.2.2. *Consider the domain Ω discretised into a grid (with triangle or rectangle elements) with nodes at x_k. An example of a possible choice for β_k is continuous piecewise linear/bilinear hat functions with respect to this grid. In this case, m_k is simply the value of the model m at the point x_k, i.e., $m_k = m(x_k)$. Each β_k is supported locally on the elements that have x_k among its nodes.*

We denote the PDE solution operator of the forward problem $\mathscr{S}_{\boldsymbol{m},\omega}$, i.e., for a given source term f, we write the wavefield u as

$$u = \mathscr{S}_{\boldsymbol{m},\omega} f$$

such that u is the solution of the following Helmholtz problem

$$u = \mathscr{S}_{\boldsymbol{m},\omega} f \quad \Longleftrightarrow \quad \begin{cases} -(\Delta + \omega^2 m)u &= f \quad \text{on} \quad \Omega \\ (\partial/\partial n - \mathrm{i}\omega)u &= 0 \quad \text{on} \quad \partial\Omega \end{cases} \tag{2.2.5}$$

where the boundary condition here is the impedance boundary condition, similar to (D-8), but with the assumption that $m \equiv 1$ on $\partial\Omega$. This assumption is made for convenience and could be removed. When $f \in L^2$ and $m \in L^\infty$, then Proposition 3.4.18 states that a solution u to the problem (2.2.5) exists and is unique. We note here that in Section 2.4.4 we prove results under certain assumptions on the solution operator $\mathscr{S}_{\boldsymbol{m},\omega}$, stated in Assumption 2.4.10.

Later in the thesis, we are also interested in the *adjoint problem*. We write the adjoint solution operator as $\mathscr{S}^*_{\boldsymbol{m},\omega}$, so that

$$v = \mathscr{S}^*_{\boldsymbol{m},\omega} g \quad \Longleftrightarrow \quad \begin{cases} -(\Delta + \omega^2 m)v &= g \quad \text{on} \quad \Omega \\ (\partial/\partial n + \mathrm{i}\omega)v &= 0 \quad \text{on} \quad \partial\Omega \end{cases} \tag{2.2.6}$$

We can write (2.2.5) and (2.2.6) in weak form, as

$$a_{\boldsymbol{m},\omega}(\mathscr{S}_{\boldsymbol{m},\omega} f, w) = (f, w), \quad \text{for all} \quad w \in H^1(\Omega), \tag{2.2.7}$$

and

$$a^*_{\boldsymbol{m},\omega}(\mathscr{S}^*_{\boldsymbol{m},\omega} g, w) = (g, w), \quad \text{for all} \quad w \in H^1(\Omega), \tag{2.2.8}$$

where, using $\overline{\ }$ to denote complex conjugate,

$$a_{\boldsymbol{m},\omega}(u, w) = \int_\Omega \left(\nabla u \cdot \nabla \overline{w} - \omega^2 m\, u\overline{w}\right) - \mathrm{i}\omega \int_{\partial\Omega} u\overline{w} \tag{2.2.9}$$

and

$$a^*_{\boldsymbol{m},\omega}(v, w) = \int_\Omega \left(\nabla v \cdot \nabla \overline{w} - \omega^2 m\, v\overline{w}\right) + \mathrm{i}\omega \int_{\partial\Omega} v\overline{w}. \tag{2.2.10}$$

Here $(\cdot,\cdot)$ denotes the $L^2(\Omega)$ inner product defined by,

$$(v, w) = \int_\Omega v\overline{w}.$$

and the definitions of a $L^2(\Omega)$ space and Hilbert space $H^1(\Omega)$ are, respectively,

$$L^2(\Omega) = \left\{ v : \int_\Omega |v|^2 < \infty \right\}, \qquad H^1(\Omega) = \left\{ v : v \in L^2(\Omega) \text{ and } \nabla v \in L^2(\Omega) \right\}.$$

Later in this thesis we use the following definition of the local weighted L^2 inner product

$$(v, w)_{\beta_k} = \int_\Omega \beta_k v\overline{w}.$$

The definitions (2.2.9) and (2.2.10) imply that

$$\overline{a_{\boldsymbol{m},\omega}(z, w)} = a^*_{\boldsymbol{m},\omega}(w, z)\ , \quad \text{for all} \quad w, z \in H^1(\Omega). \tag{2.2.11}$$

Proposition 2.2.3. ***Adjoint Property:*** *For all $f, g \in L^2(\Omega)$, all $\boldsymbol{m} \in \mathbb{R}^M$, and $\omega \in \mathcal{W}$,*

$$(f, \mathscr{S}^*_{\boldsymbol{m},\omega} g) = \overline{(g, \mathscr{S}_{\boldsymbol{m},\omega} f)} = (\mathscr{S}_{\boldsymbol{m},\omega} f, g) \tag{2.2.12}$$

Proof. Recall definitions (2.2.5) and (2.2.6) that $u = \mathscr{S}_{\boldsymbol{m},\omega} f$ and $v = \mathscr{S}^*_{\boldsymbol{m},\omega} g$. Then,

$$\begin{aligned}(f, \mathscr{S}^*_{\boldsymbol{m},\omega} g) &\underset{(2.2.7)}{=} a_{\boldsymbol{m},\omega}(u, \mathscr{S}^*_{\boldsymbol{m},\omega} g) \underset{(2.2.11)}{=} \overline{a^*_{\boldsymbol{m},\omega}(\mathscr{S}^*_{\boldsymbol{m},\omega} g, u)} \underset{(2.2.8)}{=} \overline{(g, u)} = \overline{(g, \mathscr{S}_{\boldsymbol{m},\omega} f)} \\ &= (\mathscr{S}_{\boldsymbol{m},\omega} f, g),\end{aligned}$$

where the final equality is by definition of the inner product. ∎

We are often interested specifically in the PDE problem where $f = \delta(x - s) = \delta_s$ (i.e., a point source at s representing a typical seismic survey source). For a given source position $s \in \mathcal{S}$, frequency $\omega \in \mathcal{W}$ and model parameters $\boldsymbol{m} \in \mathbb{R}^M$, we define the wavefield $u = u(\boldsymbol{m}, \omega, s)$ to be the solution of the equation

$$u(\boldsymbol{m}, \omega, s) = \mathscr{S}_{\boldsymbol{m},\omega} \delta_s \quad \Longleftrightarrow \quad \begin{cases} -(\Delta + \omega^2 m)u &= \delta_s \quad \text{on} \quad \Omega \\ (\partial/\partial n - \mathrm{i}\omega)u &= 0 \quad \text{on} \quad \partial\Omega \end{cases} \tag{2.2.13}$$

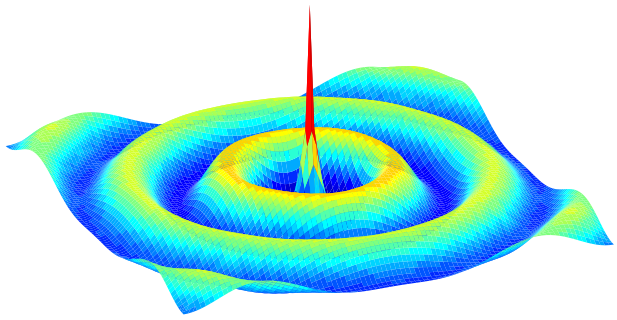


Figure 2.2.1: *Illustration of a typical numerical solution to (2.2.13), at a chosen frequency, in a homogeneous medium, with a point source in the centre of the domain.*

Remark 2.2.4. ***Continuity of*** *u:* *We make a short note here on the continuity of u defined by (2.2.5) where m is defined by (2.2.4), using some results detailed later in the thesis. Proposition 3.4.18 states that for a source $f \in L^2(\Omega)$, the Helmholtz problem has a unique solution $u \in H^1(\Omega)$. Rearranging the PDE (2.2.5) as*

$$\Delta u = -\omega^2 m u - f,$$

and assuming $m \in L^\infty(\Omega)$, then using Proposition 3.4.19 (i), we find that $u \in H^2(\Omega)$. By Proposition 3.4.16, u is therefore continuous. Hence, if the basis functions in (2.2.4) are such that $m \in L^\infty(\Omega)$, then u is continuous.

The same results hold for v defined by (2.2.6). We present results on the continuity of u generated by a point source in Section 3.4.3.

2.2.3 Discretisation

Wave equations, along with appropriate boundary conditions, usually do not have exact solutions, and so approximations to the solutions are commonly found using numerical methods. Solving a wave equation numerically involves the discretization of the wave equation and its corresponding boundary conditions. Common numerical techniques applied to discretise the wave equations in the geophysics community include finite differences (FD) and finite elements (FE), in either the time or frequency domain.

We focus on the frequency domain wave equation (2.2.1). Discretising (2.2.1) results in a system of linear equations, where the right hand side is the discretised source, and the solution is the discretised wavefield,

$$A(\boldsymbol{m}, \omega)\mathbf{u}(\boldsymbol{m}, s, \omega) = \mathbf{f}(s, \omega). \tag{2.2.14}$$

Defining N as the total number of discretisation points, the complex matrix $A \in \mathbb{C}^{N\times N}$ is the discretised wave-equation operator. The form of A depends on the method used to discretise the PDE. When using FE or FD, A is sparse. The vector $\mathbf{f}(s, \omega) \in \mathbb{C}^{N\times 1}$ is the discretised source term for the sth source at angular frequency ω (the source term may also be independent of frequency). The wavefield due to this source is denoted $\mathbf{u}(\boldsymbol{m}, s, \omega) \in \mathbb{C}^{N\times 1}$ and is a function of the model (some physical subsurface parameter) $\boldsymbol{m} \in \mathbb{R}^{M\times 1}$, where M is the number of parameters in the model. Note that the wavefield $\mathbf{u}$ and model $\boldsymbol{m}$ can describe two-dimensional or three-dimensional quantities, but are arranged into a vector here by giving the nodes an appropriate ordering. Scalar quantities u_i or m_i denotes the ith entry in the discretised wavefield and model vector respectively.

Remark 2.2.5. *As noted in [129], for certain forward problems, there may be more than one parameter class for $\boldsymbol{m}$, (i.e., more than one physical property that we are considering, e.g., wave velocities, density, and Lamé parameters). In this multi-class case we would have $\boldsymbol{m} \in \mathbb{R}^{(M\times N_{par})\times 1}$, where $N_{par} \in \mathbb{N}$ is the number of parameter classes.*

Note that equation (2.2.14) is a general representation of the discretised version (either FD or FE) of both the elastic and acoustic wave equations. This thesis generally presents theorems and lemmas with the wave equation in its continuous formulation, however we use its discrete form when discussing computations. In addition, the discrete form is often the form used in the geophysics literature so including the discrete version of formulas is useful for the geophysics applications of the work in this thesis.

We provide example discretisation schemes for the Helmholtz equation in Appendix E. A finite difference scheme is described since we use a finite difference discretisation in our experiments (as provided in [180]). Since the finite element method is closely connected to the weak formulation (for example (2.2.7)), a finite element discretisation is also presented. Furthermore, we show how finite differences and finite elements are related in Appendix E. Alternative approaches to the discretisation of the Helmholtz equation, and analysis of these discretisations, can be found in, for example, [69, Chapter 2], [186], [67], and [142].

2.2.4 FWI Objective Function

The FWI objective function (or misfit function) involves a measure of the misfit between observed/measured data $\mathbf{d}$, and modelled data $\mathbf{d}^{mod}$ in a chosen norm.

We recall that the modelled data $\mathbf{d}^{mod}$ is the data that comes from solving the forward problem (equation (2.1.1)) using the current estimate of the true model. More specifically, the modelled data is computed by solving the forward problem to find the wavefield, and sampling the values of the wavefield at the sensor positions. Therefore, before we define the objective function, we must discuss the sampling of the wavefield required to compute the $\mathbf{d}^{mod}$.

Restriction Operator: To mathematically formulate the process of computing modelled data, we introduce a *restriction operator*, or sampling operator, $\mathcal{R}$. The restriction operator $\mathcal{R}(\mathcal{P})$ applied to the wavefield u evaluates the wavefield at the sensors, i.e.,

$$\mathcal{R}(\mathcal{P})u = \begin{pmatrix} u(p_1) \\ . \\ . \\ . \\ u(p_{N_r}) \end{pmatrix}. \tag{2.2.15}$$

The modelled data is then,

$$\mathbf{d}^{mod}(\boldsymbol{m}, \mathcal{P}, s, \omega) = \mathcal{R}(\mathcal{P})u(\boldsymbol{m}, \omega, s). \tag{2.2.16}$$

In this thesis, it will be convenient to write the restriction operator applied to the function u on Ω in the following form

$$\mathcal{R}(\mathcal{P})u = \begin{pmatrix} (u, \delta_{p_1}) \\ . \\ . \\ . \\ \left(u, \delta_{p_{N_r}}\right) \end{pmatrix} \tag{2.2.17}$$

where δ_{p_j} is the delta function centred at the sensor position p_j. Note that in general we use the notation $(\cdot, \cdot)$ to denote the L^2 inner product, but in (2.2.17) we have to extend this notation to allow the 'generalised function' δ_{p_j} to be included, i.e.,

$$(u, \delta_{p_j}) = \int_\Omega u \delta_{p_j} = u(p_j).$$

Remark 2.2.4 gives conditions under which the point evaluation of u, and hence $\mathcal{R}(\mathcal{P})$, is well-defined.

The adjoint of the restriction operator is then defined by

$$\mathcal{R}(\mathcal{P})^* \boldsymbol{z} = \sum_{j=1}^{N_r} \delta_{p_j} z_j, \quad \text{for} \quad \boldsymbol{z} \in \mathbb{C}^{N_r}, \tag{2.2.18}$$

and it can be checked that

$$\langle \mathcal{R}(\mathcal{P})u, \boldsymbol{z}\rangle = (u, \mathcal{R}(\mathcal{P})^*\boldsymbol{z}), \tag{2.2.19}$$

where $\langle\cdot,\cdot\rangle$ denotes the Euclidean inner product on $\mathbb{C}^{N_r}$.

The restriction of the wavefield to the sensor positions is similarly represented by delta functions in the literature, for example in [71], [190], and [63]. We note that alternative definitions of the restriction operator also exist in FWI. In particular, in the FWI problem with Cauchy data, as described for example in [69, Chapter 7], the restriction operator is defined as the evaluation of both the wavefield (specifically the pressure wavefield in this case) and the normal derivative of the wavefield at the set of sensor positions.

Remark 2.2.6. ***Discretisation of Restriction Operator:*** *In the discretised FWI objective function (with forward problem of the form (2.2.14)), we can think of the coordinates of all positions in space at which the sensors are located as a matrix $\boldsymbol{P} \in \mathbb{R}^{N_r\times d}$, where N_r is the number of sensors and d is the number of spatial dimensions we are working in, i.e., the lth row contains the coordinates of the lth sensor. The set of sensors positions $\mathcal{P}$ is such that*

$$\mathcal{P} = \{\boldsymbol{P}_l \,:\, l = 1,\dots,N_r\} \subseteq \Omega,$$

where $\boldsymbol{P}_l$ is the lth row, containing the coordinates of the lth sensor. We also write the vector $\boldsymbol{p} \in \mathbb{R}^{dN_r\times 1}$ to mean the concatenated columns of $\boldsymbol{P}$. Note that $\boldsymbol{P}$ and $\boldsymbol{p}$ are just different ways of expressing the same information and here we just write it both ways to help understanding.

The discrete restriction operator, which we denote $R(\boldsymbol{p})$, is an $N_r \times N$ matrix (recall N denotes the number of discretisation nodes) where the lth row of $R(\boldsymbol{p})$ is the restriction operator for the lth sensor. As the sensor positions are not tied to the discretisation nodes, each row of $R(\boldsymbol{p})$ represents the action of an interpolation operator. The restriction operator, $R(\boldsymbol{p})$, can then be applied to the wavefield $\boldsymbol{u}$ to evaluate the wavefield at the sensor locations $\boldsymbol{p}$.

We now give an example of a particular choice of the restriction operator. Consider a two dimensional domain $\Omega = [0, L] \times [0, L]$, so that $d = 2$. Suppose the wavefield is computed at points on the uniform grid

$$X_{i,j} = (x_i, z_j) = (ih, jh), \qquad i, j = 0,\dots,n-1,$$

where $h = L/(n-1)$, and $n^2 = N$. That is, $u_{i,j} = u(X_{i,j})$ for each $i, j = 0,\dots,n-1$. We define a bilinear interpolant on Ω by

$$\mathcal{I}\boldsymbol{u} = \sum_{i,j=0}^{n-1} \gamma_{i,j} u_{i,j}$$

where $\gamma_{i,j}$ are the piecewise bilinear basis functions with

$$\gamma_{i,j}(X_{i',j'}) = \begin{cases} 1 & \text{if } i' = i, j' = j \\ 0 & \text{otherwise.} \end{cases}$$

Then, for $l = 1, \ldots, N_r$, the lth row of $R(\boldsymbol{p})$ is given by the entries

$$\{\gamma_{i,j}(\boldsymbol{P}_l) : i, j = 0, \ldots, n-1\}. \tag{2.2.20}$$

Note the entries in (2.2.20) are ordered according to the chosen ordering of the discretisation nodes (e.g., lexicographic ordering). By (2.2.20), the matrix $R(\boldsymbol{p})$ is very sparse, since the only non-zero values on the lth row correspond to the local nodes surrounding the sensor positions $\boldsymbol{P}_l$. If $\boldsymbol{P}_l$ is a grid point $X_{i,j}$, then we simply have that $(R(\boldsymbol{p})\boldsymbol{u})_l = u_{i,j}$. In practice, a piecewise bilinear interpolant is sufficient to perform FWI. However, we will see in Chapter 3 that it is not sufficient for sensor placement optimisation.

We now discuss the discrete version of the adjoint of the restriction operator. The adjoint of the restriction operator in the discrete world is simply the transpose of the interpolant matrix, i.e., $R(\boldsymbol{p})^* = R(\boldsymbol{p})^T \in \mathbb{R}^{N \times N_r}$ The discrete analogue of (2.2.19) is then

$$\langle R(\boldsymbol{p})\boldsymbol{u}, \boldsymbol{z}\rangle = \langle \boldsymbol{u}, R(\boldsymbol{p})^T \boldsymbol{z}\rangle,$$

where the inner product on the left hand side is in $\mathbb{C}^{N_r}$ while on the right hand side it is in $\mathbb{C}^N$. The term $R(\boldsymbol{p})^T\boldsymbol{z}$ is a sparse $N \times 1$ vector, such that $\boldsymbol{z}$ is extended to the entire computational grid. The positions of the only non-zero entries of the resulting vector correspond to the positions of the sensors, i.e., the jth entry of $\boldsymbol{z}$ is positioned at the jth sensor location.

Misfit: Returning to the continuous (non-discrete formulation), we now define the *misfit* (or residual) between the observed and modelled data. The misfit is

$$\boldsymbol{\varepsilon}(\boldsymbol{m}, \mathcal{P}, \omega, s) = \mathbf{d}(\mathcal{P}, \omega, s) - \mathcal{R}(\mathcal{P})u(\boldsymbol{m}, \omega, s) \quad \in \mathbb{C}^{N_r}, \tag{2.2.21}$$

where $\mathbf{d} \in \mathbb{C}^{N_r \times 1}$ is the data recorded at the sensors and is a function of sensor positions. The misfit may also be thought of as the residual wavefield at receiver points.

The FWI objective function is constructed as a sum of misfits over all sources and frequencies.

Definition 2.2.7. ***Full Waveform Inversion Objective Function:*** *The FWI misfit/objective function is*

$$\phi(\boldsymbol{m}, \mathcal{P}) := \frac{1}{2}\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}} ||\boldsymbol{\varepsilon}(\boldsymbol{m}, \mathcal{P}, \omega, s)||_2^2, \tag{2.2.22}$$

where the misfit $\boldsymbol{\varepsilon}$ is given by (2.2.21).

Computing the objective function involves the following steps. Firstly, the appropriate

wave equation is solved to obtain the solution u. Then, this solution is restricted to the receiver locations with the restriction operator (2.2.16). Finally, the misfit is computed in the chosen norm.

The objective function can be defined in different ways by changing the norm used in (2.2.22). We have chosen the traditional least-squares functional based on the L_2 norm as it has been widely used and remains the most popular. However, we note that alternative norms do exist and have shown some advantages, such as increased robustness to noise and amplitude outliers, for example the L_1 norm in [34], or norms that provide a compromise between L_1 and L_2 norm, for example in [52] and [83]. The objective function may also be defined with a weighting operator applied to the misfit, in order to give more or less influence to certain data. Some examples of where a weighted objective function has been considered in the literature are [141], [14], [98], [40] and [55].

2.2.5 FWI Solution

The reconstruction of the subsurface parameters is expressed as the following minimisation problem

$$\widehat{\boldsymbol{m}}(\mathcal{P}) \in \operatorname*{argmin}_{\boldsymbol{m}} \phi(\boldsymbol{m}, \mathcal{P}), \tag{2.2.23}$$

where the FWI solution $\widehat{\boldsymbol{m}}$ is one of possibly many solutions, due to the ill-posedness of the FWI problem. In this section we discuss how the solution of the FWI problem can be found in practice, as well as some theoretical aspects of the solution.

Solving the FWI Problem in Practice The solution to the FWI problem is found by iteratively minimising the objective function ϕ using an optimisation method. Successive updates of the subsurface model are found so that the modelled data can eventually match the physical measurements (within some tolerance). Appendix F reviews some standard optimisation methods that are commonly used in FWI. Quasi-Newton methods are among the most popular optimisation methods in FWI. In particular, the limited-memory BFGS (L-BFGS) method has been proven to be very effective for large-scale applications [128], and is therefore used extensively to solve the FWI problem, see for instance [42], [155] and [54]. Although gradient-based optimisation methods are generally favoured in FWI for their computational efficiency, methods that use Hessian information have been seen to be more robust. For example, [128] suggests to use the Truncated Newton method, and [150] recommends the use of the Gauss-Newton or Newton method.

Theoretical Considerations The FWI problem (with objective function in Definition 2.2.7) is related to the following inverse problem at the level of the PDE: given certain 'information' about the forward operator (i.e., (2.2.3)), find m. One choice of information about the forward operator is the Dirichlet-to-Neumann (DtN) map at certain frequencies (i.e., the map that takes Dirichlet data, solves the Helmholtz problem with that data, and then returns the normal derivative on the boundary of

the domain). An alternative choice of information about the forward operator is the so-called Cauchy data, i.e., pairs of Dirichlet and Neumann data on the boundary such that there exists a Helmholtz solution with these boundary values. For both these choices of information, one can additionally assume that one only has information on a subset of the boundary, i.e., one only has the local Dirichlet-to-Neumann map or local (as opposed to global) Cauchy data. See [10, Section 1] for the definition of the local Dirichlet-to-Neumann map, and [11, Section 2.3] for the definition of local Cauchy data.

Stability results for the inverse problem for the Helmholtz equation with a DtN map as the data are presented in [24] for a piecewise constant representation of the model, and extended to a piecewise linear representation of the model in [69, Section 3.4.2]. Stability results are provided in [11, Theorem 2] for the inverse problem for the Helmholtz equation with Cauchy data.

The notion of uniqueness of the FWI solution is also an important consideration. Uniqueness guarantees that one solution exists, i.e., it means there is only one subsurface that can produce the observed data. In general, FWI does not have a unique solution. It is argued in [121] that the acoustic FWI problem is intrinsically non-unique. There are some more general discussions of uniqueness for the seismic problem in [178]. Incorporating prior information, in the form of a regularisation term, can help to resolve the issue of non-uniqueness.

2.3 Gradient of ϕ with respect to $\boldsymbol{m}$

Many of the optimisation methods for FWI detailed in Appendix F require the computation of the gradient of the FWI objective function ϕ with respect to the model $\boldsymbol{m}$. Therefore, in this section, we derive the gradient of ϕ, for the forward problem (2.2.5) with the model defined by (2.2.4).

Computing the gradient requires taking partial derivatives of (2.2.22) with respect to each model parameter m_k, for $k = 1, \ldots, M$. By directly differentiating ϕ in (2.2.22) with respect to the kth model element, we find that the kth entry of the FWI gradient of is

$$\frac{\partial \phi}{\partial m_k}(\boldsymbol{m}, \mathcal{P}) \;=\; -\Re\left[\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left\langle \mathcal{R}(\mathcal{P})\frac{\partial u}{\partial m_k}(\boldsymbol{m},\omega,s), \boldsymbol{\varepsilon}(\boldsymbol{m},\mathcal{P},\omega,s)\right\rangle\right], \quad k = 1,\ldots,M. \tag{2.3.1}$$

The term $\partial u/\partial m_k$ is the derivative of the wavefield with respect to the kth model parameter.[1] To find a PDE that $\partial u/\partial m_k$ satisfies, we differentiate Equation (2.2.5) (or

[1] Strictly speaking, u is understood as the solution of the variational problem (2.2.7), and so $\partial u/\partial m_k$ is understood as the solution of the variational problem (2.2.7) differentiated with respect to m_k (which is the weak form of (2.3.2)). Under the assumptions described in Remark 2.2.4, the restriction operator applied to u (and also $\partial u/\partial m_k$) returns a vector in $\mathbb{C}^{N_r}$, and thus the derivatives of ϕ with respect to m_k are then standard derivatives.

equivalently (2.2.13)) with respect to m_k, to obtain

$$\begin{aligned} -(\Delta + \omega^2 m)\frac{\partial u}{\partial m_k}(\boldsymbol{m}, \omega, s) &= \omega^2 \beta_k u(\boldsymbol{m}, \omega, s) \qquad \text{on } \Omega, \\ \left(\frac{\partial}{\partial n} - i\omega\right)\frac{\partial u}{\partial m_k}(\boldsymbol{m}, \omega, s) &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{2.3.2}$$

Therefore, by our definition of the solution operator (2.2.5), we can write (2.3.2) as

$$\frac{\partial u}{\partial m_k}(\boldsymbol{m}, \omega, s) = \omega^2 \mathscr{S}_{\boldsymbol{m},\omega}(\beta_k u(\boldsymbol{m}, \omega, s)). \tag{2.3.3}$$

There are M such equations to be solved to find $\partial u/\partial m_k$ for each k (plus an extra one initially to find u that forms the right hand side of (2.3.3)). This means that for each parameter in the model, there is a corresponding PDE to be solved, for each source and each frequency, to compute the gradient via formula (2.3.1). However, it is possible to formulate the gradient in a way that the number of PDE solves becomes independent of the number of parameters in the model. This method of making the number of PDE solves to be independent of the number of model parameters is called the *Adjoint-State Method*. We derive this alternative formulation of the gradient in Theorem 2.3.1. We note that this result is also presented in [150] for the general discrete formulation of FWI.

Theorem 2.3.1. ***Adjoint-State Method for the FWI Gradient***
For each $\boldsymbol{m}, \mathcal{P}$ and each $k = 1, \ldots, M$,

$$\frac{\partial \phi}{\partial m_k}(\boldsymbol{m}, \mathcal{P}) \;=\; -\Re\left(\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}} \omega^2 \left(u(\boldsymbol{m}, \omega, s), \lambda(\boldsymbol{m}, \omega, s, \mathcal{P})\right)_{\beta_k}\right), \tag{2.3.4}$$

where, for each frequency ω and source s, $\lambda(\boldsymbol{m}, \omega, s, \mathcal{P})$ is the solution to the adjoint wave equation

$$\lambda(\boldsymbol{m}, \omega, s, \mathcal{P}) = \mathscr{S}^*_{\boldsymbol{m},\omega}\left(\mathcal{R}(\mathcal{P})^* \boldsymbol{\varepsilon}(\boldsymbol{m}, \omega, s, \mathcal{P})\right). \tag{2.3.5}$$

Proof. To simplify the notation we assume there is only one source s and one frequency ω, and we suppress their dependence here. Thus we can write (2.3.1) more simply as

$$\frac{\partial \phi}{\partial m_k}(\boldsymbol{m}, \mathcal{P}) \;=\; -\Re\left\langle \mathcal{R}(\mathcal{P})\frac{\partial u}{\partial m_k}(\boldsymbol{m}), \boldsymbol{\varepsilon}(\boldsymbol{m}, \mathcal{P})\right\rangle, \quad k = 1, \ldots, M.$$

We apply the identity (2.2.19) to rewrite this as

$$\frac{\partial \phi}{\partial m_k}(\boldsymbol{m}, \mathcal{P}) \;=\; -\Re\left(\frac{\partial u}{\partial m_k}(\boldsymbol{m}), \mathcal{R}(\mathcal{P})^* \boldsymbol{\varepsilon}(\boldsymbol{m}, \mathcal{P})\right), \quad k = 1, \ldots, M. \tag{2.3.6}$$

Substituting (2.3.3) into (2.3.6) gives

$$\frac{\partial \phi}{\partial m_k}(\boldsymbol{m}, \mathcal{P}) \;=\; -\Re\left(\omega^2 \mathscr{S}_{\boldsymbol{m}}(\beta_k u(\boldsymbol{m})), \mathcal{R}(\mathcal{P})^* \boldsymbol{\varepsilon}(\boldsymbol{m}, \mathcal{P})\right), \quad k = 1, \ldots, M,$$

Using (2.2.12) gives, for $k = 1, \ldots, M$,

$$\begin{aligned}\frac{\partial \phi}{\partial m_k}(\boldsymbol{m}, \mathcal{P}) &= -\omega^2 \,\Re\,(\,\beta_k u(\boldsymbol{m}), \mathscr{S}^*_{\boldsymbol{m}}(\mathcal{R}(\mathcal{P})^* \boldsymbol{\varepsilon}(\boldsymbol{m}, \mathcal{P}))\,), \\ &= -\omega^2 \,\Re\,(\,u(\boldsymbol{m}), \mathscr{S}^*_{\boldsymbol{m}}(\mathcal{R}(\mathcal{P})^* \boldsymbol{\varepsilon}(\boldsymbol{m}, \mathcal{P}))\,)_{\beta_k}, \\ &= -\omega^2 \,\Re\,(\,u(\boldsymbol{m}), \lambda(\boldsymbol{m}, \mathcal{P})\,)_{\beta_k}, \end{aligned} \tag{2.3.7}$$

where the adjoint variable λ satisfies (2.3.5). We have proved the one source, one frequency case, which extends easily to many sources and many frequencies by including a sum over sources and frequencies in (2.3.7). ∎

Theorem 2.3.1 shows that, instead of solving $M + 1$ PDEs per source per frequency to find the gradient, as in (2.3.1) and (2.3.3), we can directly compute the gradient with only **2 PDE solves per source per frequency** using (2.3.4). These PDEs are the forward wave equation (2.2.5) and the *adjoint wave equation* (2.3.5). We note that a review of the adjoint-state method, and an alternative derivation of the gradient from Lagrangian persepctive, are provided in [143].

The computation of the gradient of ϕ is summarised in the following steps:

(i) Solve the forward problem (2.2.5) to find the wavefield u.

(ii) Compute the *adjoint wavefield* λ by solving the adjoint wave-equation (2.3.5).

(iii) Compute the weighted inner-product between the forward and adjoint field, as in (2.3.7).

(iv) This weighted inner-product is repeated for each frequency and each source, the sum of each result is formed and the real part of the result is taken to give (2.3.4).

Remark 2.3.2. ***Number of PDE Solves in FWI with a Gradient-Based Optimisation Method:*** *We previously stated that the computation of the FWI gradient requires 2 PDE solves, for each source and each frequency. When using a gradient-based optimisation method, as outlined in Appendix F, this results in 2 PDE solves, for each source, each frequency and each iteration of optimisation method, leading to a total of $2N_sN_\omega N_{it}$ PDE solves, where N_s and N_ω are defined in Table 2.2.1 and N_{it} denotes the number of iterations of the chosen optimisation method until convergence. Furthermore, we see in Appendix F that many optimisation methods also involve computing a step size. We note here that computing each step size may require several evaluations of the gradient, depending on the type of the line search used. Therefore, in practice, there are much more than $2N_sN_\omega N_{it}$ PDE solves required during a gradient-based optimisation approach to FWI.*

The following two remarks discuss the discretised version of the formulae presented in this section.

Remark 2.3.3. ***Discretisation of (2.3.3):*** *In the discretised version of the problem (with forward equation (2.2.14)), the derivative of the discretised wavefield with respect to the kth model parameter, $\partial \boldsymbol{u}/\partial m_k$, satisfies*

$$A(\boldsymbol{m},\omega)\frac{\partial \boldsymbol{u}(\boldsymbol{m},s,\omega)}{\partial m_k} = -\left(\frac{\partial A(\boldsymbol{m},\omega)}{\partial m_k}\right)\boldsymbol{u}(\boldsymbol{m},s,\omega) \qquad k=1,\ldots,M. \tag{2.3.8}$$

The solution $\partial \boldsymbol{u}/\partial m_k$ to the wave equation (2.3.8) is often known in the geophysics literature as the 'partial derivative wavefield from the kth node' (see [150] for example). The right-hand side of (2.3.8) involves partial derivatives of the wave operator matrix A with respect to the model, $\partial A/\partial m_k$. This $\partial A/\partial m_k$ term depends on the specific details of the matrix A (i.e., the type of wave equation and the method used for discretisation). In the simplest case, consider the Helmholtz wave equation (2.2.5), with finite difference discretisation where the model $\boldsymbol{m}$ has one coefficient per node, for all N nodes. Then, the model coefficients only appear in the diagonal of the discretization matrix A. The differentiation with respect to one of those coefficients m_k gives a matrix with only one non-zero value at the diagonal position (k,k) and this non-zero value is $-\omega^2$ (if we assume the boundary condition term doesn't depend on the model). The resulting right-hand-side of (2.3.8) will then be highly local,

$$\left(\frac{\partial A}{\partial m_k}\boldsymbol{u}\right)_j = \begin{cases} 0 & \text{if } j \neq k, \\ -\omega^2 u_k & \text{if } j = k. \end{cases} \tag{2.3.9}$$

In the case of finite element discretisation, the term $\partial A/\partial m_k$ depends on the basis functions (denoted β_k in (2.2.4)). In the rest of this thesis, when referring to the discretisation, we do not specify the details of the term $\partial A/\partial m_k$ so formulae are kept general for any choice discretisation.

Remark 2.3.4. ***Discretisation of the FWI Gradient:*** *With the forward problem given by (2.2.14), the discretised version of (2.3.1) is*

$$\frac{\partial \phi(\boldsymbol{m},\boldsymbol{p})}{\partial m_k} = -\Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[\left(R(\boldsymbol{p})\frac{\partial \boldsymbol{u}(\boldsymbol{m},s,\omega)}{\partial m_k}\right)^* \boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{p},s,\omega)\right]\right\} \tag{2.3.10}$$

and the discretised versions of (2.3.4) and (2.3.5) are

$$\frac{\partial \phi(\boldsymbol{m},\boldsymbol{p})}{\partial m_k} = \Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[\left(\frac{\partial A(\boldsymbol{m},\omega)}{\partial m_k}\boldsymbol{u}(\boldsymbol{m},s,\omega)\right)^* \boldsymbol{\lambda}(\boldsymbol{m},\boldsymbol{p},s,\omega)\right]\right\}, \tag{2.3.11}$$

$$A(\boldsymbol{m},\omega)^*\boldsymbol{\lambda}(\boldsymbol{m},\boldsymbol{p},s,\omega) = R(\boldsymbol{p})^*\boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{p},s,\omega). \tag{2.3.12}$$

respectively. We note that when indicating the dependence on the sensor positions in these discretised formulae, we write the vector of sensor coordinates $\boldsymbol{p}$ rather than the set $\mathcal{P}$ to make the distinction between the restriction operator in the continuous setting, $\mathcal{R}(\mathcal{P})$, and in the discrete setting, $R(\boldsymbol{p})$, clear.

2.4 Hessian of ϕ with respect to $\boldsymbol{m}$

In this section we derive formulae for the computation of the Hessian and Hessian-vector products, discuss the cost of computation in terms of the number of PDE solves, investigate the structure of the Hessian, and derive novel results about the positive-definitiveness of the Hessian and the uniqueness of the solution to the FWI problem.

2.4.1 Hessian Computation

We can compute the entries of the Hessian by differentiating (2.3.1), with respect to the model parameter m_j, for $j = 1, \ldots, M$,

$$(H(\boldsymbol{m}, \mathcal{P}))_{j,k} = \frac{\partial^2 \phi}{\partial m_j \partial m_k}(\boldsymbol{m}, \mathcal{P}) = \Re\left(\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[\left\langle \mathcal{R}(\mathcal{P})\frac{\partial u}{\partial m_k}(\boldsymbol{m},\omega,s), \mathcal{R}(\mathcal{P})\frac{\partial u}{\partial m_j}(\boldsymbol{m},\omega,s)\right\rangle - \left\langle \mathcal{R}(\mathcal{P})\frac{\partial^2 u}{\partial m_k \partial m_j}(\boldsymbol{m},\omega,s), \boldsymbol{\varepsilon}(\boldsymbol{m},\mathcal{P},\omega,s)\right\rangle\right]\right). \tag{2.4.1}$$

The Hessian is written as the sum of two parts - a term involving first order derivatives of the wavefield u with respect to the model, which we denote $H^{(1)}$, and a term involving second order derivatives of u with respect to the model, which we denote $H^{(2)}$. We write the Hessian as a sum as

$$H(\boldsymbol{m}, \mathcal{P}) = H^{(1)}(\boldsymbol{m}, \mathcal{P}) + H^{(2)}(\boldsymbol{m}, \mathcal{P}), \tag{2.4.2}$$

where

$$\left(H^{(1)}(\boldsymbol{m}, \mathcal{P})\right)_{j,k} = \Re\left(\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[\left\langle \mathcal{R}(\mathcal{P})\frac{\partial u}{\partial m_k}(\boldsymbol{m},\omega,s), \mathcal{R}(\mathcal{P})\frac{\partial u}{\partial m_j}(\boldsymbol{m},\omega,s)\right\rangle\right)\right] \tag{2.4.3}$$

$$\left(H^{(2)}(\boldsymbol{m}, \mathcal{P})\right)_{j,k} = \Re\left(\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[-\left\langle \mathcal{R}(\mathcal{P})\frac{\partial^2 u}{\partial m_k \partial m_j}(\boldsymbol{m},\omega,s), \boldsymbol{\varepsilon}(\boldsymbol{m},\mathcal{P},\omega,s)\right\rangle\right]\right) \tag{2.4.4}$$

for $k, j = 1, .., M$. The term $H^{(1)}$ is often referred to as the approximate, or Gauss-Newton, Hessian. We note that the Hessian is symmetric and so the indices j and k in the above formulae can be swapped.

To assemble $H^{(1)}$ (2.4.3) we require the first-order derivatives of the wavefield with respect to each component of $\boldsymbol{m}$, i.e., we need

$$\frac{\partial u}{\partial m_k}, \qquad \text{for} \quad k = 1, \ldots, M. \tag{2.4.5}$$

The computation of (2.4.5) involves M PDE solves for each source s and each frequency ω, where the PDE is defined by (2.3.3). The term $H^{(2)}$ (2.4.4) involves the second derivative of the wavefield with respect to $\boldsymbol{m}$,

$$\frac{\partial^2 u}{\partial m_k \partial m_j}, \qquad \text{for} \quad k, j = 1, \ldots, M. \tag{2.4.6}$$

We differentiate (2.3.2) with respect to m_j to obtain the following PDE for (2.4.6),

$$\begin{aligned} -(\Delta + \omega^2 m)\frac{\partial^2 u}{\partial m_k \partial m_j}(\boldsymbol{m},\omega,s) &= \omega^2\left(\beta_k \frac{\partial u}{\partial m_j}(\boldsymbol{m},\omega,s) + \beta_j \frac{\partial u}{\partial m_k}(\boldsymbol{m},\omega,s)\right) \text{ on } \Omega \\ \left(\frac{\partial}{\partial n} + i\omega\right)\frac{\partial^2 u}{\partial m_k \partial m_j}(\boldsymbol{m},\omega,s) &= 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{on } \partial\Omega. \end{aligned} \tag{2.4.7}$$

Then, by our definition of the solution operator (2.2.5),

$$\frac{\partial^2 u}{\partial m_k \partial m_j}(\boldsymbol{m},\omega,s) = \omega^2 \mathscr{S}_{\boldsymbol{m},\omega}\left(\beta_k \frac{\partial u}{\partial m_j}(\boldsymbol{m},\omega,s) + \beta_j \frac{\partial u}{\partial m_k}(\boldsymbol{m},\omega,s)\right), \tag{2.4.8}$$

for $k, j = 1, \ldots, M$. Therefore, if we wanted to compute (2.4.6), this would require the solution to M^2 PDEs, on top of the M PDEs required to compute $\partial u/\partial m_k$ for each k, as part of the right-hand side of (2.4.8). Therefore, assuming the wavefield u has already been computed for each source and frequency, then computing the $H^{(1)}$ term given by (2.4.3) requires M solves for each source and frequency, and computing the $H^{(2)}$ term given by (2.4.4) term would require an additional M^2 solves for each source and frequency. However, in Theorem 2.4.1, we show that by applying the adjoint-state method, we do not require all M^2 solves (per soruce per frequency) to assemble $H^{(2)}$, and instead only require the same M PDE solves (per soruce per frequency) required to assemble $H^{(1)}$. This result is also described in [150] for the general discrete formulation.

Theorem 2.4.1. ***Adjoint-State Method for $H^{(2)}$***
For each $\boldsymbol{m}, \mathcal{P}$ and each $k, j = 1, \ldots, M$,

$$\begin{aligned} \left(H^{(2)}(\boldsymbol{m},\mathcal{P})\right)_{k,j} = -\Re\Bigg(\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}} \omega^2 \Bigg[&\left(\frac{\partial u}{\partial m_j}(\boldsymbol{m},\omega,s), \lambda(\boldsymbol{m},\mathcal{P},\omega,s)\right)_{\beta_k} \\ &+ \left(\frac{\partial u}{\partial m_k}(\boldsymbol{m},\omega,s), \lambda(\boldsymbol{m},\mathcal{P},\omega,s)\right)_{\beta_j}\Bigg]\Bigg) \end{aligned} \tag{2.4.9}$$

where λ is defined in (2.3.5).

Proof. To simplify the notation we assume there is only one source s and one frequency ω, and we suppress their dependence in the notation. Therefore we can write $H^{(2)}$ in (2.4.4) as

$$\left(H^{(2)}(\boldsymbol{m},\mathcal{P})\right)_{k,j} = \Re\left[-\left\langle \mathcal{R}(\mathcal{P})\frac{\partial^2 u}{\partial m_k \partial m_j}(\boldsymbol{m}), \boldsymbol{\varepsilon}(\boldsymbol{m},\mathcal{P})\right\rangle\right],$$

By (2.2.19), the above can be written as

$$(H^{(2)}(\boldsymbol{m},\mathcal{P}))_{k,j} = -\Re\left(\frac{\partial^2 u}{\partial m_k \partial m_j}(\boldsymbol{m}), \mathcal{R}(\mathcal{P})^* \boldsymbol{\varepsilon}(\boldsymbol{m},\mathcal{P})\right). \tag{2.4.10}$$

Substituting (2.4.8) into (2.4.10) gives

$$
\begin{aligned}
(H^{(2)}(\boldsymbol{m},\mathcal{P}))_{k,j} &= -\Re\left(\omega^2\mathscr{S}_{\boldsymbol{m}}\left(\beta_k\frac{\partial u}{\partial m_j}(\boldsymbol{m})+\beta_j\frac{\partial u}{\partial m_k}(\boldsymbol{m})\right),\mathcal{R}(\mathcal{P})^*\boldsymbol{\varepsilon}(\boldsymbol{m},\mathcal{P})\right)\\
&= -\omega^2\Re\left(\left(\beta_k\frac{\partial u}{\partial m_j}(\boldsymbol{m})+\beta_j\frac{\partial u}{\partial m_k}(\boldsymbol{m})\right),\mathscr{S}^*_{\boldsymbol{m}}\mathcal{R}(\mathcal{P})^*\boldsymbol{\varepsilon}(\boldsymbol{m},\mathcal{P})\right).
\end{aligned}
$$

Using the definition of the adjoint variable λ in (2.3.5), the above can be rewritten as

$$
\begin{aligned}
(H^{(2)}(\boldsymbol{m},\mathcal{P}))_{k,j} &= -\omega^2\Re\left(\left(\beta_k\frac{\partial u}{\partial m_j}(\boldsymbol{m})+\beta_j\frac{\partial u}{\partial m_k}(\boldsymbol{m})\right),\lambda(\boldsymbol{m},\mathcal{P})\right)\\
&= -\omega^2\Re\left[\left(\frac{\partial u}{\partial m_j}(\boldsymbol{m},\omega,s),\lambda(\boldsymbol{m},\mathcal{P})\right)_{\beta_k}+\left(\frac{\partial u}{\partial m_k}(\boldsymbol{m}),\lambda(\boldsymbol{m},\mathcal{P})\right)_{\beta_j}\right].
\end{aligned}
$$

Thus the one source, one frequency case has been proved, which can be extended to the multi-source, multi-frequency case to give (2.4.9). ∎

Theorem (2.4.1) shows that $H^{(2)}$ can be written as a sum of weighted inner-products, involving the first-order derivative of the wavefield (2.4.5) only, and does not require the computation of second-order derivative of the wavefield (2.4.6).

Using Theorem (2.4.1), the $(k,j)th$ element of the full Hessian H (2.4.1) can be written as

$$
\begin{aligned}
(H(\boldsymbol{m},\mathcal{P}))_{k,j} = \ \Re\Bigg(\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\Bigg[&\left\langle\mathcal{R}(\mathcal{P})\frac{\partial u}{\partial m_k}(\boldsymbol{m},\omega,s),\mathcal{R}(\mathcal{P})\frac{\partial u}{\partial m_j}(\boldsymbol{m},\omega,s)\right\rangle\\
&-\omega^2\left(\left(\frac{\partial u}{\partial m_j}(\boldsymbol{m},\omega,s),\lambda(\boldsymbol{m},\mathcal{P},\omega,s)\right)_{\beta_k}\right.\\
&\left.\left.+\left(\frac{\partial u}{\partial m_k}(\boldsymbol{m},\omega,s),\lambda(\boldsymbol{m},\mathcal{P},\omega,s)\right)_{\beta_j}\right)\Bigg]\Bigg)
\end{aligned}
\tag{2.4.11}
$$

for $k,j=1,\ldots,M$.

The computation of the Hessian of ϕ is summarised by the following steps:

(i) Solve the forward problem (2.2.5) to find u.

(ii) Compute the adjoint wavefield λ by solving the adjoint wave equation (2.3.5).

(iii) Compute the first-order partial derivative matrix (2.4.5) by solving M wave-equations (2.3.3).

(iv) Repeat for all sources and frequencies, and assemble the Hessian according to (2.4.11).

In total, to compute the Hessian, for each source and each frequency, requires **M+2 PDE solves per source per frequency**, 2 of which would have already computed in practice to find the gradient. We note that even though the computational requirements are reduced due to the adjoint-state method, the cost of explicitly computing the elements of the Hessian is unreasonably high, as the value of M in real seismic imaging applications can be huge. In addition to the extensive computational demands, the large dimension of the Hessian and the memory restrictions related to this means that the computation and storage of the full Hessian is often unrealistic for optimisation in FWI, and optimisation methods that only require the computation of the gradient of ϕ are usually chosen. However, Pratt et al. [150] and Métivier et al. [129] argue that there is important physical information contained in the Hessian which should not be ignored, and numerical results suggest that making use of the Hessian can play an important role in the convergence when solving the FWI problem. In Section 2.4.2 we discuss how Hessian-based optimisation methods can be used in FWI without having to explicitly compute or store the Hessian.

Remark 2.4.2. ***Discretisation of the FWI Hessian:*** *With the forward problem (2.2.14), the discretised version of (2.4.1) is*

$$(H(\boldsymbol{m},\boldsymbol{p}))_{k,j} = \frac{\partial^2\phi(\boldsymbol{m},\boldsymbol{p})}{\partial m_j \partial m_k} = \Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[\left(R(\boldsymbol{p})\frac{\partial \boldsymbol{u}(\boldsymbol{m},s,\omega)}{\partial m_k}\right)^*\left(R(\boldsymbol{p})\frac{\partial \boldsymbol{u}(\boldsymbol{m},s,\omega)}{\partial m_j}\right)\right.\right. \\ \left.\left. - \left(R(\boldsymbol{p})\frac{\partial^2 \boldsymbol{u}(\boldsymbol{m},s,\omega_\alpha)}{\partial m_k \partial m_j}\right)^* \boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{p},s,\omega)\right]\right\}, \quad k,j = 1,\dots M, \tag{2.4.12}$$

with $H^{(1)}$ and $H^{(2)}$ defined as

$$(H^{(1)}(\boldsymbol{m},\boldsymbol{p}))_{k,j} = \Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[\left(R(\boldsymbol{p})\frac{\partial \boldsymbol{u}(\boldsymbol{m},s,\omega)}{\partial m_k}\right)^*\left(R(\boldsymbol{p})\frac{\partial \boldsymbol{u}(\boldsymbol{m},s,\omega)}{\partial m_j}\right)\right]\right\}, \tag{2.4.13}$$

$$(H^{(2)}(\boldsymbol{m},\boldsymbol{p}))_{k,j} = \Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[-\left(R(\boldsymbol{p})\frac{\partial^2 \boldsymbol{u}(\boldsymbol{m},s,\omega)}{\partial m_k \partial m_j}\right)^* \boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{p},s,\omega)\right]\right\}. \tag{2.4.14}$$

The discretised version of (2.4.11) is

$$(H(\boldsymbol{m},\boldsymbol{p}))_{k,j} = \frac{\partial^2\phi(\boldsymbol{m},\boldsymbol{p})}{\partial m_k \partial m_j} \tag{2.4.15}$$

$$= \Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[\left(R(\boldsymbol{p})\frac{\partial \boldsymbol{u}(\boldsymbol{m},s,\omega)}{\partial m_j}\right)^*\left(R(\boldsymbol{p})\frac{\partial \boldsymbol{u}(\boldsymbol{m},s,\omega)}{\partial m_k}\right)\right.\right. \\ \left.\left. + \left(\frac{\partial A(\boldsymbol{m},\omega)}{\partial m_k}\frac{\partial \boldsymbol{u}(\boldsymbol{m},\omega,s)}{\partial m_j} + \frac{\partial A(\boldsymbol{m},\omega)}{\partial m_j}\frac{\partial \boldsymbol{u}(\boldsymbol{m},\omega,s)}{\partial m_k}\right)^* \boldsymbol{\lambda}(\boldsymbol{m},\boldsymbol{p},\omega,s)\right]\right\}, \tag{2.4.16}$$

for $k,j = 1,\dots M$. Sometimes a term involving the second derivative of A with respect to the model is included in the defintion of the Hessian; for example, see [150]. The

exact details of the second derivative term depend on the type of forward model and the exact details of the discretisation, but in most cases is very sparse. We don't include this second derivative term here as for our specific example of A in Section 2.2.3, the second derivative of A with respect to the model would be zero.

2.4.2 Hessian-Vector Products

Although we have reduced the full cost of computing the Hessian in Theorem 2.4.1 to $M+2$ PDE solves for each source and frequency, this is still an excessive number number of PDEs to be solved in practice. However, instead of computing the Hessian explicitly for each step of an optimisation method (such as Newton's method) and solving the Hessian system directly, methods can be used to provide an iterative solution to the Hessian system that avoid the explicit computation of the Hessian and require only Hessian-vector products. An example of an optimisation method that requires only Hessian-vector products is the Truncated Newton method, which used in FWI in [129].

For each source and each frequency, the computation of Hessian-vector products involves only 4 PDE solves, a reduction of $M-2$ PDE solves from computing the Hessian itself. In Theorem 2.4.3 we derive how Hessian-vector products can be computed in only 4 PDE solves using the second-order adjoint-state method.

Theorem 2.4.3. ***Second-Order Adjoint-State Method for Hessian-Vector Products***
For an arbitrary vector $\widetilde{\boldsymbol{m}} \in \mathbb{R}^{M\times 1}$, where M is the number of components in the model, the kth component of the Hessian-vector product $H\widetilde{\boldsymbol{m}}$ is

$$\begin{aligned}(H(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}})_k \;= \Re\Bigg\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\omega^2\Big[&(u(\boldsymbol{m},\omega,s), z(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P},\omega,s))_{\beta_k} \\ &-(v(\boldsymbol{m},\widetilde{\boldsymbol{m}},\omega,s),\lambda(\boldsymbol{m},\mathcal{P},\omega,s))_{\beta_k}\Big]\Bigg\}, \end{aligned} \tag{2.4.17}$$

where u is defined by (2.2.5), λ is defined by (2.3.5), and

$$v(\boldsymbol{m},\widetilde{\boldsymbol{m}},\omega,s) := \omega^2\,\mathscr{S}_{\boldsymbol{m},\omega}\left(\widetilde{m}u(\boldsymbol{m},\omega,s)\right), \tag{2.4.18}$$

$$z(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P},\omega,s) := \mathscr{S}^*_{\boldsymbol{m},\omega}\left(\mathcal{R}(\mathcal{P})^*\mathcal{R}(\mathcal{P})v(\boldsymbol{m},\widetilde{\boldsymbol{m}},\omega,s) - \omega^2\widetilde{m}\lambda(\boldsymbol{m},\mathcal{P},\omega,s)\right) \tag{2.4.19}$$

where $\widetilde{m} = \sum_k \widetilde{m}_k\beta_k$.

Proof. To simplify the notation we assume there is only one source s and one frequency ω, and we suppress their dependence in the notation.
First note that by the definition of $\widetilde{m}$, the linearity of $\mathscr{S}_{\boldsymbol{m}}$, and definition (2.3.3), the

PDE for v (2.4.18) can be written as

$$v(\boldsymbol{m}, \widetilde{\boldsymbol{m}}) = \sum_j \widetilde{m}_j \mathscr{S}_{\boldsymbol{m}}(\omega^2 \beta_j u(\boldsymbol{m})) = \sum_j \widetilde{m}_j \frac{\partial u}{\partial m_j}(\boldsymbol{m}). \tag{2.4.20}$$

By the one-source one-frequency version of (2.4.11), we can write the Hessian-vector product as

$$\begin{aligned}(H(\boldsymbol{m}, \mathcal{P})\widetilde{\boldsymbol{m}})_k &= \sum_j (H(\boldsymbol{m}, \mathcal{P}))_{k,j} \widetilde{m}_j \\ &= \sum_j \left(\Re \left[\left\langle \mathcal{R}(\mathcal{P}) \frac{\partial u}{\partial m_k}(\boldsymbol{m}), \mathcal{R}(\mathcal{P}) \frac{\partial u}{\partial m_j}(\boldsymbol{m}) \right\rangle \right.\right. \\ &\qquad \left.\left. -\omega^2 \left(\left(\frac{\partial u}{\partial m_j}(\boldsymbol{m}), \lambda(\boldsymbol{m}, \mathcal{P}) \right)_{\beta_k} + \left(\frac{\partial u}{\partial m_k}(\boldsymbol{m}), \lambda(\boldsymbol{m}, \mathcal{P}) \right)_{\beta_j} \right) \right] \right) \widetilde{m}_j.\end{aligned}$$

Then, using the definitions (2.4.20) and (2.3.3), we obtain

$$\begin{aligned}(H(\boldsymbol{m}, \mathcal{P})\widetilde{\boldsymbol{m}})_k &= \Re \left[\left(\frac{\partial u}{\partial m_k}(\boldsymbol{m}), \mathcal{R}(\mathcal{P})^* \mathcal{R}(\mathcal{P}) v(\boldsymbol{m}, \widetilde{\boldsymbol{m}}) \right) \right. \\ &\qquad \left. -\omega^2 \left(v(\boldsymbol{m}, \widetilde{\boldsymbol{m}}), \lambda(\boldsymbol{m}, \mathcal{P}) \right)_{\beta_k} - \omega^2 \left(\frac{\partial u}{\partial m_k}(\boldsymbol{m}), \widetilde{m} \lambda(\boldsymbol{m}, \mathcal{P}) \right) \right] \\ &= \Re \left[\left(\frac{\partial u}{\partial m_k}(\boldsymbol{m}), \mathcal{R}(\mathcal{P})^* \mathcal{R}(\mathcal{P}) v(\boldsymbol{m}, \widetilde{\boldsymbol{m}}) - \omega^2 \widetilde{m} \lambda(\boldsymbol{m}, \mathcal{P}) \right) - \omega^2 \left(v(\boldsymbol{m}, \widetilde{\boldsymbol{m}}), \lambda(\boldsymbol{m}, \mathcal{P}) \right)_{\beta_k} \right] \\ &= \Re \left[\left(\omega^2 \mathscr{S}_{\boldsymbol{m}}(\beta_k u(\boldsymbol{m})), \mathcal{R}(\mathcal{P})^* \mathcal{R}(\mathcal{P}) v(\boldsymbol{m}, \widetilde{\boldsymbol{m}}) - \omega^2 \widetilde{m} \lambda(\boldsymbol{m}, \mathcal{P}) \right) - \omega^2 \left(v(\boldsymbol{m}, \widetilde{\boldsymbol{m}}), \lambda(\boldsymbol{m}, \mathcal{P}) \right)_{\beta_k} \right].\end{aligned}$$

By the definition of $\mathbf{z}$ in (2.4.19), the above can be written as

$$(H(\boldsymbol{m}, \mathcal{P})\widetilde{\boldsymbol{m}})_k = \omega^2 \Re \left[\left(u(\boldsymbol{m}), z(\boldsymbol{m}, \widetilde{\boldsymbol{m}}, \mathcal{P}) \right)_{\beta_k} - \left(v(\boldsymbol{m}, \widetilde{\boldsymbol{m}}), \lambda(\boldsymbol{m}, \mathcal{P}) \right)_{\beta_k} \right],$$

as required. The result extends to the case with many sources and many frequencies by including a sum over sources and frequencies. ∎

The two additional PDE solves (2.4.18) and (2.4.19) are written in discrete form in [127, Equation 12] and [127, Equation 15] respectively.

Remark 2.4.4. ***Discrete Version of Hessian-Vector Product Formulae:*** *The jth element of the Hessian-vector product, for $j = 1, \ldots, M$, is*

$$\begin{aligned}(H(\boldsymbol{m}, \boldsymbol{p})\widetilde{\boldsymbol{m}})_j = \Re \left\{ \sum_{s \in \mathcal{S}} \sum_{\omega \in \mathcal{W}} \left[- \left(\frac{\partial A(\boldsymbol{m}, \omega)}{\partial m_j} \boldsymbol{u}(\boldsymbol{m}, s, \omega), \boldsymbol{z}(\boldsymbol{m}, \boldsymbol{p}, \omega) \right) \right.\right. \\ \left.\left. + \left(\frac{\partial A(\boldsymbol{m}, \omega)}{\partial m_j} \boldsymbol{v}(\boldsymbol{m}, s, \omega), \boldsymbol{\lambda}(\boldsymbol{m}, \boldsymbol{p}, s, \omega) \right) \right] \right\}\end{aligned} \tag{2.4.21}$$

where the wavefield $\boldsymbol{u}$ *and adjoint wavefield* $\boldsymbol{\lambda}$ *are given discretely as the solutions to the equations (2.2.14) and (2.3.12) respectively, and* $\boldsymbol{v}$ *and* $\boldsymbol{z}$ *are defined as the solutions to the following PDEs,*

$$A(\boldsymbol{m},\omega)\boldsymbol{v}(\boldsymbol{m},s,\omega) = -\sum_k \widetilde{m}_k \left(\frac{\partial A(\boldsymbol{m},\omega)}{\partial m_k}\boldsymbol{u}(\boldsymbol{m},s,\omega)\right) \tag{2.4.22}$$

$$A(\boldsymbol{m},\omega)^*\boldsymbol{z}(\boldsymbol{m},\boldsymbol{p},s,\omega) = R(\boldsymbol{p})^*R(\boldsymbol{p})\boldsymbol{v}(\boldsymbol{m},\omega) + \sum_k \widetilde{m}_k \left(\frac{\partial A(\boldsymbol{m},\omega)}{\partial m_k}^*\boldsymbol{\lambda}(\boldsymbol{m},\boldsymbol{p},s,\omega)\right). \tag{2.4.23}$$

A term involving the second derivative of A with respect to the model can be included in (2.4.21) (as in [127]) but we do not include it here since the second derivative of A with respect to the model would be zero in our specific example of A in Section 2.2.3.

By Theorem 2.4.3, the Hessian-vector product can be written as the sum of 2 weighted inner products, and requires 4 PDE solves per source per frequency. The derivation in Theorem 2.4.3 describes the full Hessian. In the case where we consider the $H^{(1)}$ part of the Hessian only (which is an approximation made for the Gauss-Newton method) the kth component of the Hessian-vector product can be written as

$$(H^{(1)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}})_k = \omega^2 \Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[(u(\boldsymbol{m},\omega,s), z_1(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P},\omega,s))_{\beta_k}\right]\right\}, \tag{2.4.24}$$

where

$$z_1(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P},\omega,s) := \mathscr{S}^*_{\boldsymbol{m},\omega}(\mathcal{R}(\mathcal{P})^*\mathcal{R}(\mathcal{P})v(\boldsymbol{m},\widetilde{\boldsymbol{m}},\omega,s)). \tag{2.4.25}$$

This simplification only requires one weighted inner-product and 3 PDE solves (λ is not required).

Throughout the rest of this thesis, we often write (2.4.17) as the sum

$$(H(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}})_k = (H^{(1)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}})_k + (H^{(2)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}})_k, \tag{2.4.26}$$

where we have split the Hessian into its first-order ($H^{(1)}$) and second-order ($H^{(2)}$) parts. To split the expression (2.4.17) in the same way requires the splitting of (2.4.19) into two separate PDE solves. Note that these new PDE solves are only required for later analysis (in Section 2.4.4), and they are not necessary in practice. We define

$$(H^{(1)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}})_k = \Re\left(\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\omega^2\left[(u(\boldsymbol{m},\omega,s), z_1(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P},\omega,s))_{\beta_k}\right]\right) \tag{2.4.27}$$

$$(H^{(2)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}})_k = -\Re\left(\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\omega^2\left[(u(\boldsymbol{m},\omega,s), z_2(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P}))_{\beta_k} + (v(\boldsymbol{m},\widetilde{\boldsymbol{m}},\omega,s), \lambda(\boldsymbol{m},\mathcal{P},\omega,s))_{\beta_k}\right]\right) \tag{2.4.28}$$

and

$$z_2(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P},\omega,s) := \omega^2\mathscr{S}^*_{\boldsymbol{m},\omega}(\widetilde{m}\lambda(\boldsymbol{m},\mathcal{P},\omega,s)) \tag{2.4.29}$$

where v is defined by (2.4.18) and z_1 is defined by (2.4.25).

2.4.3 Properties of the Hessian

This section discusses the properties and structure of the Hessian, demonstrates these properties with numerical experiments, and proves some theorems about these properties.

2.4.3.1 Structure of the Hessian

The Hessian, defined in (2.4.1), is a real symmetric matrix. As noted in the previous section, the Hessian can be split into two parts, as in (2.4.2) - the matrix $H^{(1)}$, which is positive semi-definite, and the matrix $H^{(2)}$, which is generally indefinite.

When the FWI objective function is defined by (2.2.22), and the modelled data is able to reproduce the observations, then $\boldsymbol{\varepsilon} \to \mathbf{0}$ as $N_{it} \to \infty$, where N_{it} is the number of iterations of the chosen optimisation method. We can see from equation (2.4.4) that $H^{(2)}$ depends directly on $\boldsymbol{\varepsilon}$ and so, as we show later in Corollary 2.4.13, $\boldsymbol{\varepsilon} \to \mathbf{0}$ implies $||H^{(2)}||_2 \to 0$ (under certain assumptions). We also show in Theorem 2.4.7 that $H^{(1)}$ does not go to zero with $\boldsymbol{\varepsilon}$, and therefore the positive semi-definite $H^{(1)}$ term becomes dominant over the $H^{(2)}$ term as we approach the solution of the non-regularised FWI problem.

In this case where $H \to H^{(1)}$ as we get closer to the minimum $\boldsymbol{\varepsilon} = \mathbf{0}$, the Hessian becomes singular as the minimum is approached due to the rank-deficiency of $H^{(1)}$. The rank-deficiency of $H^{(1)}$ arises due to the small number of observations at receivers during acquisition, compared to the large number of model parameters that we are solving for in FWI. In fact, we show in Theorem 2.4.6 that the rank of $H^{(1)} \in \mathbb{R}^{M \times M}$ is bounded above by $2N_sN_rN_\omega$, where N_s is the number of sources, N_r is the number of receivers, N_ω is the number of frequencies, and M is the number of model parameters, which is often very large.

A simple way to overcome the issue of rank-deficiency is to include regularisation in the FWI objective function. This is discussed in detail in Section 2.4.4.

As a summary, the main points in this section are:

1. $||H^{(2)}||_2 \to 0$ as $\boldsymbol{\varepsilon} \to \mathbf{0}$ (Corollary 2.4.13).
2. $||H^{(1)}||_2 \not\to 0$ as $\boldsymbol{\varepsilon} \to \mathbf{0}$ (Theorem 2.4.7).
3. Therefore, $H \to H^{(1)}$ as $\boldsymbol{\varepsilon} \to \mathbf{0}$.
4. $H^{(1)}$ is low-rank, with a rank depending on the number of sources, sensors and frequencies (Theorem 2.4.6).

2.4.3.2 Numerical Experiments

The following experiments are included to demonstrate the properties of the Hessian outlined in Points 1 to 4 above.

Experiment 1

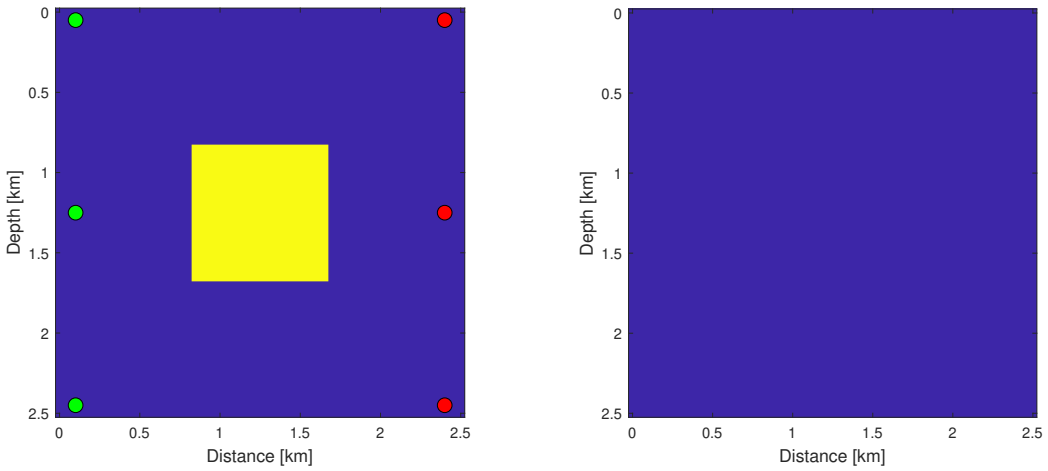
Aim: To investigate the properties of $H^{(1)}$ and $H^{(2)}$ and to demonstrate Points 1, 2

and 3 in Section 2.4.3.1 by examining the singular value and eigenvalue distributions of $H^{(1)}$ and $H^{(2)}$, both when $\boldsymbol{\varepsilon}$ is non-zero and zero.

Setup: The forward modelling in these experiments is done using the Helmholtz equation finite difference code from [180]. The code for the computation of the Hessian is original. We consider a 2500 m $\times$ 2500 m domain, discretised into a 101 $\times$ 101 grid, meaning that the model has 10201 parameters. The ground truth velocity model has a background of 2000 ms^{-1} and an inclusion of 2100 ms^{-1} (Figure 2.4.1 (a)).

The 'observed' data in this experiment is synthetic. Acquisition will be simulated from the ground truth model using a crosswell set-up, meaning that the seismic sources are located in a well/borehole, and the receivers are located in a parallel well. Acquisition is simulated using 3 equally spaced sources and 3 equally spaced sensors, and we consider 1 frequency. To avoid an inverse crime in this example, the synthetic data is computed on a different grid than that used to compute the wavefield. Specifically in this example, we compute the data on a refined grid, with grid size half that than the grid used for computing the wavefield.

Definition 2.4.5. ***Inverse Crime:*** *An inverse crime occurs when the same, or close to the same, theoretical ingredients are used to create synthetic data as well as to invert data in an inverse problem, yielding unrealistically optimistic results (this definiton comes from [185] and [100, Section 1.2]).*



(a) Exact Velocity Model (b) Smooth Background Model

Figure 2.4.1: *Models used in Section 2.4.3.2 Experiment 1. The symbol* ● *represents a source and* ● *represents a sensor.*

Part I: $\boldsymbol{\varepsilon \neq 0}$**.** The Hessian is evaluated at a smooth background model of value 2000 ms^{-1}, shown in Figure 2.4.1 (b), that is missing the inclusion of higher wavespeed in the ground truth used to compute the data (Figure 2.4.1 (a)). Therefore, the residual is not small and the second order term $H^{(2)}$ is not negligible. The eigenvalues and singular values of $H^{(1)}$ and $H^{(2)}$ are shown in Figures 2.4.2 and 2.4.3 respectively. The eigenvalue distribution of $H^{(1)}$ in Figure 2.4.2 (a) shows the rapid decay in singular

values, where only 18 of the 10201 singular values are non-zero (absolute value less than numerical zero 10^{-16}). The positive and zero eigenvalues demonstrate that $H^{(1)}$ is positive semi-definite, and the fact that there are exactly 18 non-zero eigenvalues show that $H^{(1)}$ has rank equal to $2N_rN_sN_\omega$ ($2\times$ 3 sensors $\times$ 3 sources $\times$ 1 frequency). The large gap in magnitude between the first 18 singular values and the following singular values is evident in Figure 2.4.2 (a). The condition number (ratio of largest to smallest singular value) of the Gauss-Newton Hessian $H^{(1)}$ is on the order of 10^{22}, meaning that it is severely ill-conditioned. This suggests that the inversion of $H^{(1)}$ would be unstable, and that the inclusion of $H^{(2)}$ is important. The matrix $H^{(2)}$ has no zero eigenvalues, and has both positive and negative eigenvalues (see Figure 2.4.2 (b)), which results in the indefiniteness of the Hessian. The full Hessian $H = H^{(1)} + H^{(2)}$ is full rank, and has a condition number of the order 10^7, and so is much more well-conditioned than the Gauss-Newton Hessian $H^{(1)}$.

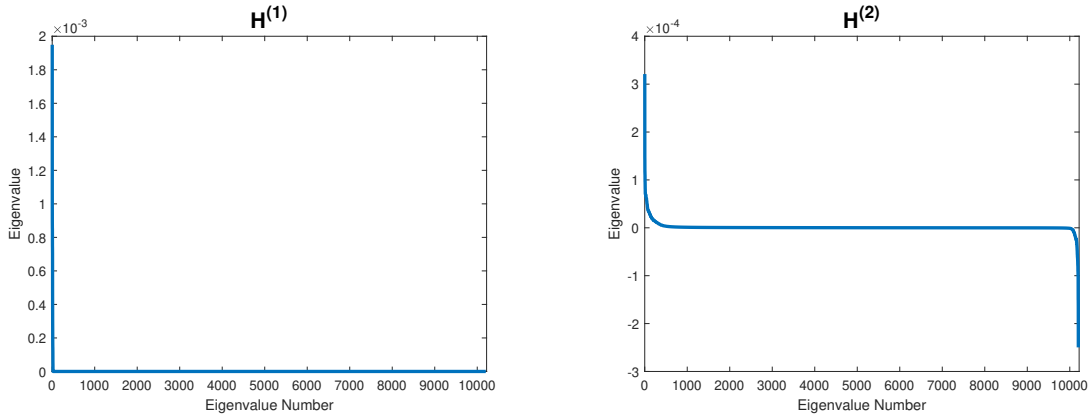


Figure 2.4.2: ***Part I:*** *Eigenvalue distribution of $H^{(1)}$ and $H^{(2)}$ in Experiment 1 when ε is non-zero.*

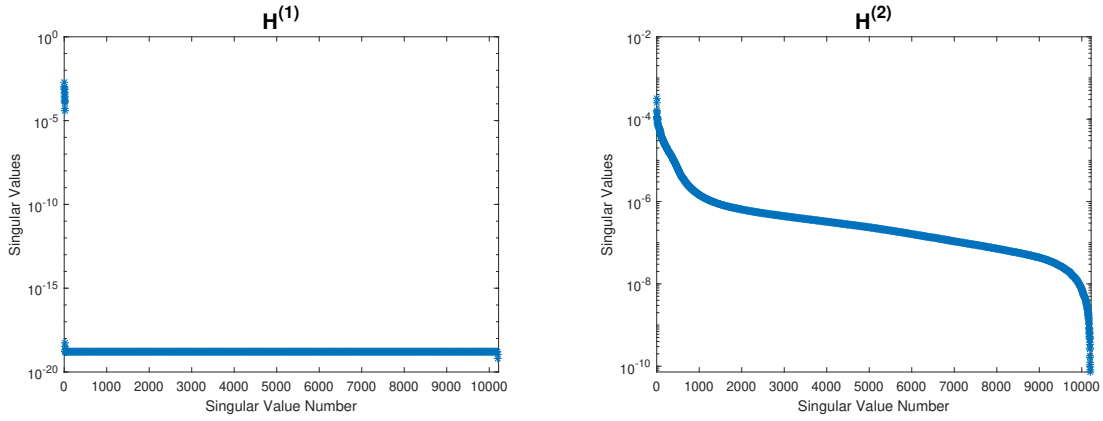


Figure 2.4.3: ***Part I:*** *Singular value distribution of $H^{(1)}$ and $H^{(2)}$ in Experiment 1 when ε is non-zero, plotted on a logarithmic scale.*

Part II: $\boldsymbol{\varepsilon \approx 0}$**.** Figures 2.4.4 and 2.4.5 show the eigenvalues and singular values of $H^{(1)}$ and $H^{(2)}$ when evaluated at a model that produces a small residual $\boldsymbol{\varepsilon}$. (This model was found using the L-BFGS method, as this method does not require the computation/inversion of the Hessian.) The properties of the $H^{(1)}$ matrix have remained the same, having only 18 positive non-zeros eigenvalues/singular values. The scale of the eigenvalues and singular values of $H^{(2)}$ have completely changed however, and are all numerically zero. This has demonstrated numerically that $||H^{(2)}||_2 \to 0$ when $\boldsymbol{\varepsilon \approx 0}$. The full Hessian now only has a rank of 18 and has a condition number of order 10^{21}.

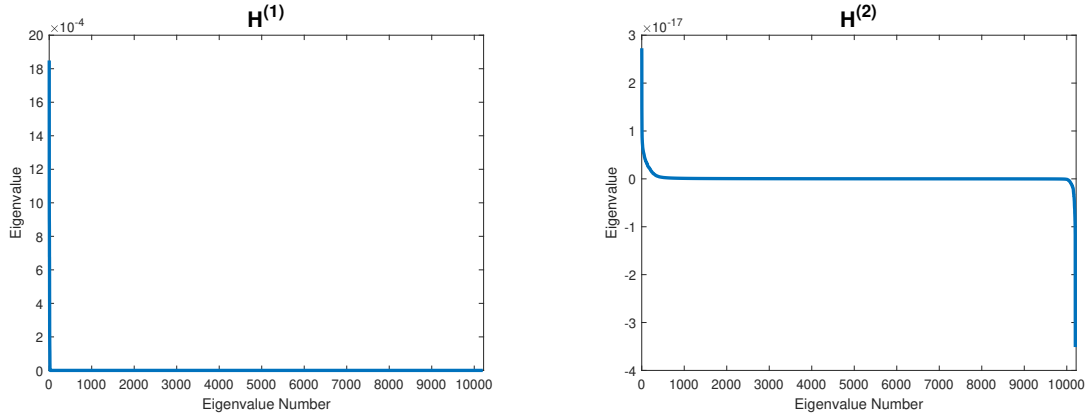


Figure 2.4.4: ***Part II:*** *Eigenvalue distribution of* $H^{(1)}$ *and* $H^{(2)}$ *in Experiment 1 when* $\boldsymbol{\varepsilon \approx 0}$.

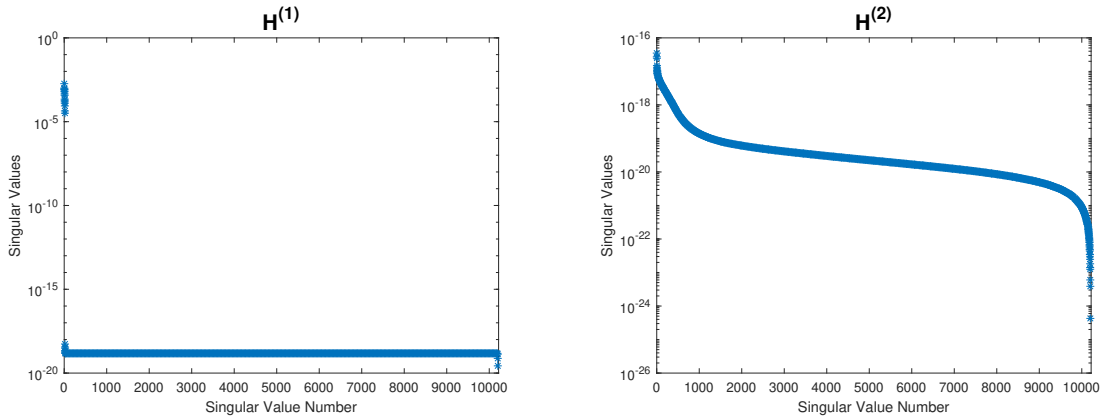


Figure 2.4.5: ***Part II:*** *Singular value distribution of* $H^{(1)}$ *and* $H^{(2)}$ *in Experiment 1 when* $\boldsymbol{\varepsilon \approx 0}$*, plotted on a logarithmic scale.*

Comparison of Part I and Part II: The full Hessian when the residual are both non-zero and zero are overlayed in Figure 2.4.6. In this example, when the residual is large, the Hessian is indefinite. When the residual becomes very small, the Hessian is positive semi-definite and extremely rank deficient, with the majority of singular values numerically zero, and the non-zero singular values coming from the $H^{(1)}$ term

only. This behaviour implies that when the term $H^{(2)}$ becomes small near the minimum $\boldsymbol{\varepsilon} = \mathbf{0}$, regularisation becomes important in ensuring the Hessian is well-conditioned. Regularisation is introduced in Section 2.4.4.

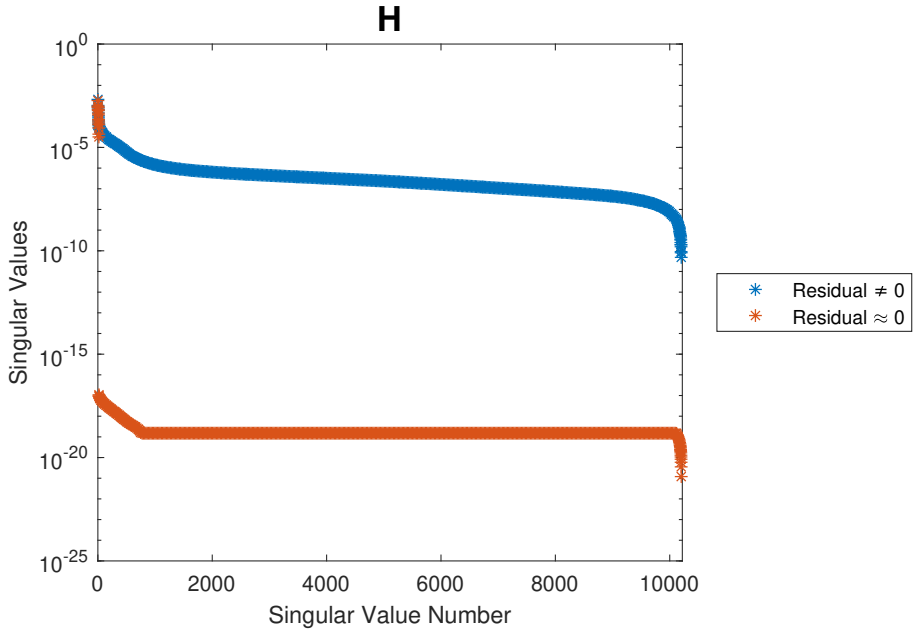


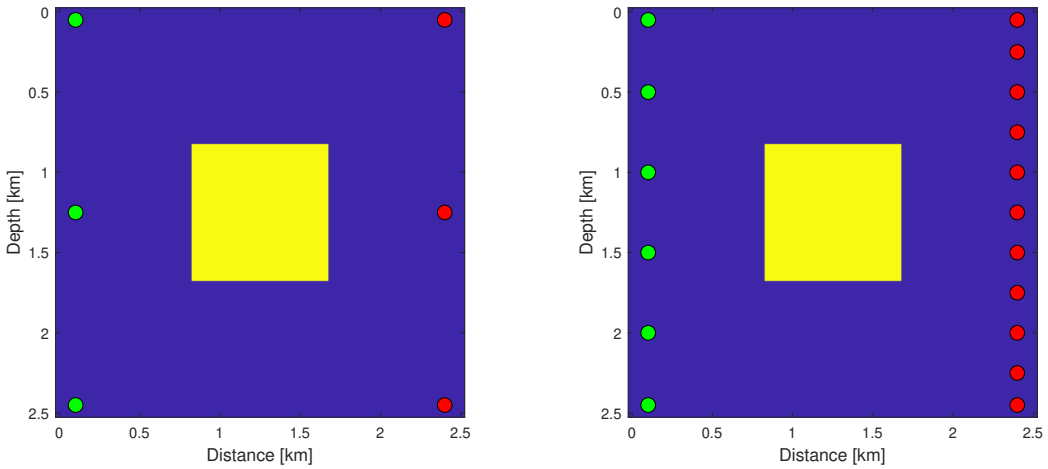
Figure 2.4.6: ***Compare Part I and Part II:*** *Singular values of* $H = H^{(1)} + H^{(2)}$ *for 2 cases: when the residual* $\boldsymbol{\varepsilon}$ *is non-zero, and when the residual* $\boldsymbol{\varepsilon}$ *is numerically zero.*

Experiment 2

Aim: The aim of this experiment is to demonstrate property 4 in Section 2.4.3.1 by examining the singular value distribution of $H^{(1)}$ for different acquisition setups.

Setup: The model we use is that shown in Figure 2.4.7. One acquisition set-up is the same as Experiment 1, with 3 sources and 3 sensors (subplot (a)). The second involves 6 sources and 11 sensors, both equally spaced (subplot (b)). We consider 1 frequency in both cases.

Results: The behaviour of the first 150 singular values is shown in Figure 2.4.8, where rapid decrease of the singular values for the 3 sources 3 sensor case can be seen. The numerous small singular values are associated with poorly illuminated model parameters [128]. More observations (i.e., more sources, sensors and frequencies) result in more non-zero singular values. The comparison between eigenvalues for the two different setups demonstrates how sufficient observations can improve the conditioning and rank of the Gauss-Newton matrix.



(a) 3 sources, 3 sensors setup (b) 6 sources, 11 sensors setup

Figure 2.4.7: *Acquisition setups used in Section 2.4.3.2 Experiment 2. The symbol ● represents a source and ● represents a sensor.*

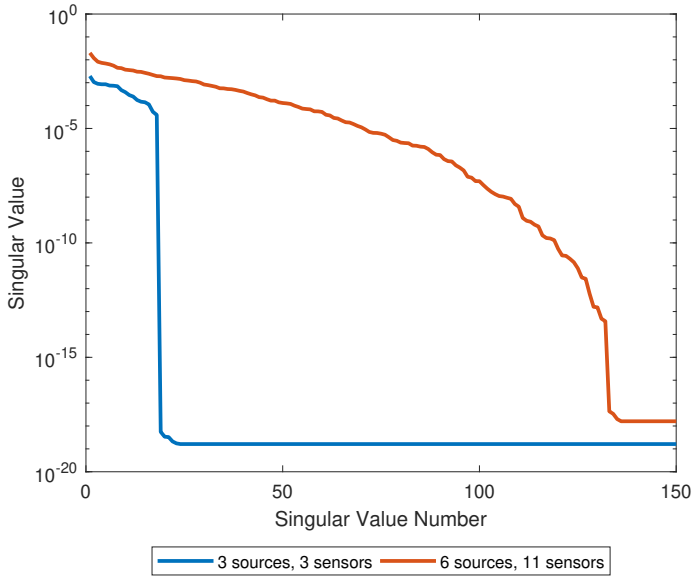


Figure 2.4.8: *150 largest singular values of $H^{(1)}$ plotted on a logarithmic scale for different acquisition set-ups. The decrease of the singular values is faster for the case with less sources and sensors which indicates a poorer conditioning of the Hessian. Specifically, there are 18 non-zero singular values for the 3 source, 3 sensor case, and 132 non-zero singular values for the 6 source, 11 sensor case.*

2.4.3.3 Results about the Structure of the Hessian

This section will present some of the main results from the discussion in Section 2.4.3.1. All theorems in this section are written in the discrete form of the problem, as this is

the natural setting when discussing the implementation of FWI. Before we present the theorems, we rewrite the Gauss-Newton Hessian $H^{(1)}$ in a form that is useful in proving our results. We begin by restating the definition of the discrete Gauss-Newton Hessian $H^{(1)}$ (2.4.13),

$$(H^{(1)}(\boldsymbol{m},\boldsymbol{p}))_{k,j} = \Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[\left(R(\boldsymbol{p})\frac{\partial \mathbf{u}(\boldsymbol{m},s,\omega)}{\partial m_j}\right)^*\left(R(\boldsymbol{p})\frac{\partial \mathbf{u}(\boldsymbol{m},s,\omega)}{\partial m_k}\right)\right]\right\},$$

where there are N_s sources in the set $\mathcal{S}$ and N_ω frequencies in the set $\mathcal{W}$. We define a new matrix $J \in \mathbb{C}^{N_r\times M}$ (the Jacobian of the residual), where the (l,k)th element is,

$$J_{l,k}(\boldsymbol{m},\boldsymbol{p},s,\omega) = \left(R(\boldsymbol{p})\frac{\partial \mathbf{u}(\boldsymbol{m},s,\omega)}{\partial m_k}\right)_l, \tag{2.4.30}$$

where $l = 1,...,N_r$. Therefore, $H^{(1)}$ can be written as,

$$H^{(1)}(\boldsymbol{m},\boldsymbol{p}) = \Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[J(\boldsymbol{m},\boldsymbol{p},s,\omega)^*J(\boldsymbol{m},\boldsymbol{p},s,\omega)\right]\right\}. \tag{2.4.31}$$

We write $H^{(1)}$ in the form (2.4.31) in the following theorems.

Theorem 2.4.6 derives an upper bound for the rank of $H^{(1)}$, and hence provides theoretical justification for the behaviour seen in Figure 2.4.8. In this proof we assume that the number of sensors is less than the number of model parameters, as this is generally true in practice.

Theorem 2.4.6. *Assuming $N_r < M$, then*

$$rank(H^{(1)}) \leq 2N_sN_rN_\omega,$$

where N_s is the number of sources, N_r is the number of sensors, N_ω is the number of frequencies, and M is the number of model parameters.

Proof. The matrix $J = J(\boldsymbol{m},\boldsymbol{p},s,\omega)$, defined by (2.4.30), has dimensions $N_r \times M$, where $N_r < M$. Therefore $rank(J) \leq N_r$.
We note the following properties for complex matrices B and C,

$$\begin{aligned} rank(B) &= rank(B^*B), \\ rank(B+C) &\leq rank(B) + rank(C). \end{aligned} \tag{2.4.32}$$

Therefore, $rank(J^*J) \leq N_r$, and

$$rank\left(\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}J^*J\right) \leq \sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}N_r = N_rN_sN_\omega. \tag{2.4.33}$$

To find $H^{(1)}$, as given by (2.4.31), we need to take the real part of (2.4.33). To find the rank of the real part, we write the complex matrix J as a sum of its real and imaginary part

$$J = J_r + i\, J_i, \tag{2.4.34}$$

with $J_r,\ J_i \in \mathbb{R}^{N_r \times M}$. Therefore

$$\begin{aligned}\Re(J^*J) = \Re\left((J_r + i\, J_i)^*(J_r + i\, J_i)\right) &= \Re\left((J_r^T - i\, J_i^T)(J_r + i\, J_i)\right) \\ &= \Re\left(J_r^T J_r - iJ_i^T J_r + iJ_r^T J_i + J_i^T J_i\right) \\ &= \Re\left(J_r^T J_r + J_i^T J_i\right) \end{aligned} \tag{2.4.35}$$

Equation (2.4.35) is the sum of two matrices with rank N_r. By the identity (2.4.32), this is less than or equal to the sum of their rank, $2N_r$. Then for all sources and receivers, by (2.4.33), $rank(H^{(1)}) \leq 2N_s N_r N_\omega$.

■

We note here an identity used by the following theorem. With J as in (2.4.34), and any real vector $\mathbf{x}$, we have that,

$$\begin{aligned}\mathbf{x}^* J^* J\mathbf{x} = \mathbf{x}^T J_r^T J_r\mathbf{x} + \mathbf{x}^T J_i^T J_i\mathbf{x} + i\left(\mathbf{x}^T J_r^T J_i\mathbf{x} - \mathbf{x}^T J_i^T J_r\mathbf{x}\right) &= \mathbf{x}^T J_r^T J_r\mathbf{x} + \mathbf{x}^T J_i^T J_i\mathbf{x} \\ = \mathbf{x}^*\Re(J^*J)\mathbf{x} \end{aligned} \tag{2.4.36}$$

We have seen in Experiment 1 in Section 2.4.3.2 that as $\boldsymbol{\varepsilon} \to \mathbf{0}$, (i.e., as we approach the solution of FWI without regularisation), $H^{(2)} \to 0$ and $H \to H^{(1)}$. Now, we want to investigate the theoretical behaviour of $H^{(1)}$. In particular, we want to examine whether $H^{(1)}$ decays like $H^{(2)}$ in any case. In Theorem (2.4.7), we provide a lower bound for $H^{(1)}$ that does not depend on $\boldsymbol{\varepsilon}$, to show that $H^{(1)}$ does not go to zero as the FWI solution is approached.

Theorem 2.4.7. *For $l = 1, ..N_r$, define $\boldsymbol{\rho}_l \in \mathbb{C}^M$ as the lth row of the Jacobian J (2.4.30), written as a column vector,*

$$\boldsymbol{\rho}_l^T(\boldsymbol{m}, \boldsymbol{p}, s, \omega) = \left\{ \left(R(\boldsymbol{p}) \frac{\partial \boldsymbol{u}(\boldsymbol{m}, s, \omega)}{\partial m_j} \right)_l : \quad for \quad j = 1, ..., M \right\}.$$

Then, we have the following lower bound for $H^{(1)}$ (2.4.31),

$$||H^{(1)}(\boldsymbol{m}, \boldsymbol{p})||_2 \geq \sum_{s \in \mathcal{S}} \sum_{\omega \in \mathcal{W}} \max_l ||\boldsymbol{\rho}_l(\boldsymbol{m}, \boldsymbol{p}, s, \omega)||_2^2.$$

Proof. We will drop the dependencies on $\boldsymbol{m}, \boldsymbol{p}, s$ and ω in this proof for simplicity. We can write

$$||J||_2^2 \;=\; \max_{\substack{x\neq 0\\ \mathbf{x}\in\mathbb{R}^M}} \frac{\mathbf{x}^* J^* J\mathbf{x}}{\mathbf{x}^*\mathbf{x}} \;=\; \max_{\substack{x\neq 0\\ x\in\mathbb{R}^M}} \frac{\mathbf{x}^* \Re(J^* J)\mathbf{x}}{\mathbf{x}^*\mathbf{x}},$$

by (2.4.36). Summing over sources and frequencies gives,

$$\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}} ||J||_2^2 \;=\; \max_{\substack{\mathbf{x}\neq 0\\ \mathbf{x}\in\mathbb{R}^M}} \frac{\mathbf{x}^* H^{(1)}\mathbf{x}}{\mathbf{x}^*\mathbf{x}} \;=\; ||H^{(1)}||_2$$

Therefore, we have that

$$||H^{(1)}||_2 \;=\; \sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}} \max_{\substack{x\neq 0\\ \mathbf{x}\in\mathbb{R}^M}} \frac{||J\mathbf{x}||_2^2}{||\mathbf{x}||_2^2}.$$

Note that

$$||J\mathbf{x}||_2^2 \;=\; \sum_{i=1}^{N_r} |\,(J\mathbf{x})_i\,|^2 \;\geq\; |\,(J\mathbf{x})_l\,|^2,$$

for any $l \in \{1, \ldots, N_r\}$. Now we choose $\mathbf{x} = \boldsymbol{\rho}_l$, for any $l \in \{1, \ldots, N_r\}$ so

$$||J\boldsymbol{\rho}_l||_2^2 \geq |\,(J\boldsymbol{\rho}_l)_l\,|^2 = |\,||\boldsymbol{\rho}_l||_2^2\,|^2 \;=\; ||\boldsymbol{\rho}_l||_2^4.$$

Therefore

$$||H^{(1)}||_2 \geq \sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}} \max_l \frac{||J\boldsymbol{\rho}_l||_2^2}{||\boldsymbol{\rho}_l||_2^2} \geq \sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}} \max_l ||\boldsymbol{\rho}_l||_2^2.$$

■

The lower bound $\boldsymbol{\rho}_l$ does not depend on $\boldsymbol{\varepsilon}$, and so as long as at least one $\boldsymbol{\rho}_l$ is not zero (which is expected), then we can guarantee that $||H^{(1)}||_2$ will remain bounded away from 0 as $\boldsymbol{\varepsilon} \to \mathbf{0}$.

Remark 2.4.8. ***Newton's method as $\boldsymbol{\varepsilon} \to \mathbf{0}$:*** *We note that as $\boldsymbol{\varepsilon} \to \mathbf{0}$ and $H \to H^{(1)}$, Newtons method approaches the Gauss Newton method. We include a theorem in Appendix G that shows that Newton's method is still consistent when $\boldsymbol{\varepsilon} \to \mathbf{0}$. In fact, there are infinitely many solutions to the Gauss-Newton system.*

Regularisation can be used to solve the issues with the rank-deficiency of the Hessian, and hence ensure a unique FWI solution. This is discussed in the following section.

2.4.4 Positive-Definiteness of the Regularised Hessian

One way of overcoming the rank-deficiency problems discussed in the previous section is to add a positive multiple of the identity to the Hessian, i.e., $H + \mu I$ with $\mu > 0$, hence ensuring the Hessian is full-rank.

This type of term can be incorporated by including a regularisation term of the form $\frac{1}{2}\mu||\boldsymbol{m}||_2^2$ to the FWI objective function (2.2.22). If prior information about the subsurface is available, a term of the form $\frac{1}{2}\mu||\boldsymbol{m} - \boldsymbol{m}_p||_2^2$ can be added, where $\boldsymbol{m}_p$ is a 'prior model', for example see [14] (which also involves a weighting matrix).

In the case where we have added a regularisation term to the objective function, we would be minimising a combination of a misfit term and regularisation term, and so $\boldsymbol{\varepsilon}$ would generally not reach zero (although it should still reduce significantly), hence preventing $H^{(2)}$ from becoming negligible during the minimisation process.

In this section, we will consider the regularised FWI problem, and derive the conditions that ensure the positive definiteness of the Hessian of the regularized problem, and discuss the connection between a positive definite Hessian and a unique solution to the FWI problem. The regularised objective function considered in this section is

$$\phi(\boldsymbol{m}, \mathcal{P}) = \frac{1}{2}\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}} \|\boldsymbol{\varepsilon}(\boldsymbol{m}, \omega, s, \mathcal{P})\|_2^2 + \frac{1}{2}\alpha\|D\boldsymbol{m}\|_2^2 + \frac{1}{2}\mu\|\boldsymbol{m}\|_2^2. \tag{2.4.37}$$

where $\mu > 0$ and $\alpha > 0$ are the regularisation parameters, and D is a discrete approximation to the first derivative. The regularisation term, $||\boldsymbol{m}||_2^2$, is referred to as the *convex term*, and the regularisation term $||D\boldsymbol{m}||_2^2$, is referred to as the *Tikhonov term*. Note that the Tikhonov term promotes smoothness in the model. The Hessian of ϕ defined in (2.4.37) is

$$H(\boldsymbol{m}, \mathcal{P}) = H^{(1)}(\boldsymbol{m}, \mathcal{P}) + H^{(2)}(\boldsymbol{m}, \mathcal{P}) + \alpha D^\top D + \mu I. \tag{2.4.38}$$

where $H^{(1)}$ and $H^{(2)}$ are given by (2.4.3) and (2.4.9) respectively.

Remark 2.4.9. ***Details of Tikhonov Matrix D:*** *We choose the matrix D to be a finite difference approximation of the first derivative. In one-dimension, this is*

$$D_1 = \frac{1}{h}\begin{pmatrix} 1 & -1 & & & & & 0 \\ & 1 & -1 & & & & \\ & & \ddots & \ddots & & & \\ & & & & 1 & -1 & \\ 0 & & & & & 1 & -1 \end{pmatrix} \in \mathbb{R}^{(n-1)\times n},$$

where h is the grid size and n is the number of discretisation nodes. In two-dimensions, a finite difference approximation of the first derivative is required in both the horizontal and vertical directions, which we denote D_x and D_z respectively. These are defined according to the lexicographic ordering of the model and the grid size in either direction. Assuming square elements and n discretisation nodes in each direction, we could define $D_x = D_1 \otimes I_n$ and $D_z = I_n \otimes D_1$, where $\otimes$ represents a Kronecker product. (Note this

definition can be extended for the case where there are a different number of discretisation nodes and different grid size in each direction). Then D in (2.4.37) can be defined as

$$D = \begin{pmatrix} D_x \\ D_z \end{pmatrix}$$

so that $\|D\boldsymbol{m}\|_2^2 = \|D_x\boldsymbol{m}\|_2^2 + \|D_z\boldsymbol{m}\|_2^2$.

We remark here that we are considering the continuous formulation of the FWI problem in this section, so that every model vector in $\mathbb{R}^M$ has a continuous representation, given by (2.2.4). For simplicity in this section, we assume only one source and one frequency. This assumption removes the sums in (2.4.37) and we avoid writing the dependencies on s and ω. The results in this section can be extended for many sources and many frequencies straightforwardly by including a sum over sources and frequencies. We also make the following assumptions.

Assumption 2.4.10.

1. *The FWI forward problem is defined by (2.2.13).*
2. *All models $\boldsymbol{m}$ lie in some set $\mathcal{M}$.*
3. *Stability for L^2 data: There exists a constant $C_0(\omega) = C_0(\omega, \mathcal{M}, \Omega)$ so that, for all $\boldsymbol{m} \in \mathcal{M}$,*

$$\max\{\|\mathscr{S}_{\boldsymbol{m},\omega} f\|_{L^2(\Omega)}, \|\mathscr{S}^*_{\boldsymbol{m},\omega} f\|_{L^2(\Omega)}\} \ \leq \ C_0(\omega)\|f\|_{L^2(\Omega)}.$$

   where $\mathscr{S}$ and $\mathscr{S}^*$ are defined by (2.2.5) and (2.2.6).

4. *Stability for point source data: There exists a constant $C_1(\omega) = C_1(\omega, \mathcal{M}, \Omega)$ so that, for all $\boldsymbol{m} \in \mathcal{M}$, and $s \in \Omega$,*

$$\max\{\|\mathscr{S}_{\boldsymbol{m},\omega}\delta_s\|_{L^2(\Omega)}, \|\mathscr{S}^*_{\boldsymbol{m},\omega}\delta_s\|_{L^2(\Omega)}\} \ \leq \ C_1(\omega)$$

5. *The wavefield is finite at the sensors positions, i.e., for all $\boldsymbol{m} \in \mathcal{M}$, there exists a constant $C_2(\omega) = C_2(\omega, \mathcal{M}, \Omega, \Omega_R)$ such that,*

$$||(\mathscr{S}_{\boldsymbol{m},\omega}\delta_s)(p)||_{\infty,\Omega_R} \leq C_2(\omega),$$

   where Ω_R is some part of the domain Ω that includes the sensors but not the sources.

6. *There exists a constant $C_3(\omega) = C_3(\omega, \mathcal{M}, \Omega)$, such that, for any sensor position p and any $f \in L^2(\Omega)$*

$$\max\{|(\mathscr{S}_{\boldsymbol{m},\omega} f)(p)|, |(\mathscr{S}^*_{\boldsymbol{m},\omega} f)(p)|\} \leq \ C_3(\omega)\|f\|_{L^2(\Omega)}.$$

The reason for considering a class of models $\mathcal{M}$ in Point 2 is that the behaviour of the Helmholtz solution operator depends strongly on the model and, in particular, whether or not the model traps rays. Indeed, if the model is trapping, then C_0 in Point 1 grows exponentially in ω; see [154], [146], [22]. Sufficient (but not necessary) conditions ensuring the models are non-trapping are given in [78, Theorems 2.5, 2.7], and thus one could consider Assumption 2.4.10 with $\mathcal{M}$ as this particular class of models. An alternative would be to restrict attention to a subset of frequencies, excluding the 'bad' frequencies through which the trapping behaviour occurs. The paper [110] proves that if a set of frequencies of arbitrarily-small measure is excluded, then C_0 grows at most polynomially in ω, i.e., the exponential growth of the solution operator associated with trapping is rare.

Proposition 2.4.11. *If a matrix A is real and symmetric, then*

$$||A||_2 = \max\{|\lambda|,\ \lambda \ \text{ eigenvalue of A}\}.$$

The Rayleigh quotient of A is bounded by

$$\lambda_{\min} \leq \frac{\langle A\boldsymbol{x}, \boldsymbol{x}\rangle}{\langle \boldsymbol{x}, \boldsymbol{x}\rangle} \leq \lambda_{\max}, \qquad \forall\, \boldsymbol{x}$$

and $\lambda_{\min}$ and $\lambda_{\max}$ are achieved for certain $\boldsymbol{x}$.

Theorem 2.4.12. ***Upper Bound on $H^{(2)}$ in terms of $\boldsymbol{\varepsilon}$:*** *If Assumptions 2.4.10 hold, then for all $\boldsymbol{m} \in \mathcal{M}$ and all $\widetilde{\boldsymbol{m}} \in \mathbb{R}^M$,*

$$\left| \frac{\left\langle H^{(2)}(\boldsymbol{m}, \mathcal{P})\widetilde{\boldsymbol{m}}, \widetilde{\boldsymbol{m}} \right\rangle}{\langle \widetilde{\boldsymbol{m}}, \widetilde{\boldsymbol{m}}\rangle} \right| \ \leq\ C(\omega)\|\boldsymbol{\varepsilon}\|_1. \tag{2.4.39}$$

where $H^{(2)}$ is defined by (2.4.9).

Proof. By Assumption 2.4.10 Point 4, $u(\boldsymbol{m})$ given by (2.2.13) satisfies

$$\|u(\boldsymbol{m})\|_{L^2(\Omega)} \leq C_1(\omega). \tag{2.4.40}$$

By the linearity of $\mathscr{S}^*_{\boldsymbol{m}}$, the definition of $\lambda(\boldsymbol{m}, \mathcal{P})$ (2.3.5), and the definition of $\mathcal{R}^*$ (2.2.18), we have

$$\lambda(\boldsymbol{m}, \mathcal{P}) \ =\ \sum_j \varepsilon_j(\boldsymbol{m}, \mathcal{P})\mathscr{S}^*_{\boldsymbol{m}}(\delta_{p_j}),$$

and so

$$\|\lambda(\boldsymbol{m},\mathcal{P})\|_{L^2(\Omega)} \;\leq\; \sum_j |\varepsilon_j(\boldsymbol{m},\mathcal{P})|\|\mathscr{S}^*_{\boldsymbol{m}}(\delta_{p_j})\|_{L^2(\Omega)} \leq C_1(\omega)\|\boldsymbol{\varepsilon}(\boldsymbol{m},\mathcal{P})\|_1. \tag{2.4.41}$$

By the definition of v in (2.4.18), the inequality (2.4.40) and Assumption 2.4.10 Point 3,

$$\|v(\boldsymbol{m},\widetilde{\boldsymbol{m}})\|_{L^2(\Omega)} \;\leq\; \omega^2 C_0(\omega)\|\widetilde{m}\,u(\boldsymbol{m})\|_{L^2(\Omega)} \leq C_1(\omega)C_0(\omega)\omega^2\|\widetilde{m}\|_{L^\infty(\Omega)}. \tag{2.4.42}$$

Similarly, by the definition of z_2 in (2.4.29) and the inequality (2.4.41),

$$\|z_2(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P})\|_{L^2(\Omega)} \;\leq\; \omega^2 C_0(\omega)\|\widetilde{m}\,\lambda(\boldsymbol{m},\mathcal{P})\|_{L^2(\Omega)} \;\leq\; \omega^2 C_0(\omega)\|\widetilde{m}\|_{L^\infty(\Omega)}C_1(\omega)\|\boldsymbol{\varepsilon}(\boldsymbol{m},\mathcal{P})\|_1, \tag{2.4.43}$$

By (2.4.28), and using that $\widetilde{m} = \sum_k \widetilde{m}_k\beta_k$, we have

$$\left\langle H^{(2)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\right\rangle \;=\; -\omega^2\Re\int_\Omega \widetilde{m}\,\left[v(\boldsymbol{m},\widetilde{\boldsymbol{m}})\,\overline{\lambda(\boldsymbol{m},\mathcal{P})} + u(\boldsymbol{m})\overline{z_2(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P})}\right].$$

Hence,

$$\begin{aligned}
&\left|\left\langle H^{(2)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\right\rangle\right| \\
&\quad\leq \omega^2\|\widetilde{m}\|_{L^\infty(\Omega)}\left[\|v(\boldsymbol{m},\widetilde{\boldsymbol{m}})\|_{L^2(\Omega)}\,\|\lambda(\boldsymbol{m},\mathcal{P})\|_{L^2(\Omega)} + \|u(\boldsymbol{m})\|_{L^2(\Omega)}\|z_2(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P})\|_{L^2(\Omega)}\right] \\
&\quad\leq 2\omega^2\|\widetilde{m}\|_{L^\infty(\Omega)}\left[\omega^2 C_1(\omega)^2C_0(\omega)\|\widetilde{m}\|_{L^\infty(\Omega)}\|\boldsymbol{\varepsilon}(\boldsymbol{m},\mathcal{P})\|_1\right].
\end{aligned}$$

Since $\|\widetilde{m}\|_{L^\infty(\Omega)} \leq \|\widetilde{\boldsymbol{m}}\|_\infty \leq \|\widetilde{\boldsymbol{m}}\|_2$, we have that

$$\left|\frac{\left\langle H^{(2)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\right\rangle}{\langle\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\rangle}\right| \;\leq\; 2\omega^4 C_1(\omega)^2C_0(\omega)\|\boldsymbol{\varepsilon}\|_1.$$

and the result follows with the definition $C(\omega) = 2\omega^4C_1(\omega)^2C_0(\omega)$. ∎

Corollary 2.4.13.

$$||H^{(2)}(\boldsymbol{m},\mathcal{P})||_2 \to 0 \quad as \quad \boldsymbol{\varepsilon}\to\boldsymbol{0}.$$

Proof. By Proposition 2.4.11 and Theorem 2.4.12, $||H^{(2)}(\boldsymbol{m},\mathcal{P})||_2 \leq C(\omega)\|\boldsymbol{\varepsilon}\|_1$. Therefore the result holds. ∎

The result of Corollary 2.4.13 relates back to the discussion in Section 2.4.3.1 and is a theoretical justification of the numerical results in Section 2.4.3.2 Experiment 1.

Theorem 2.4.12 gives an upper bound on $H^{(2)}$ in terms of the residual $\boldsymbol{\varepsilon}$, which depends

on the model. We want to extend this to find a bound on $H^{(2)}$ that is independent of the model, so that later theorems can make conclusions for all models $\boldsymbol{m} \in \mathcal{M}$. To extend the result in (2.4.39), we first find an upper bound for the $\boldsymbol{\varepsilon}$ term in Lemma 2.4.14.

Lemma 2.4.14. ***Upper Bound on*** $\boldsymbol{\varepsilon}$*:* *If Assumption 2.4.10 holds, then*

$$||\boldsymbol{\varepsilon}(\boldsymbol{m}, \mathcal{P})||_1 \leq ||\boldsymbol{d}||_1 + N_r C_2(\omega).$$

Proof. By the triangle inequality, and definition (2.2.22),

$$||\boldsymbol{\varepsilon}(\boldsymbol{m}, \mathcal{P})||_1 \leq ||\mathbf{d}||_1 + ||\mathcal{R}(\mathcal{P})u(\boldsymbol{m})||_1 = ||\mathbf{d}||_1 + \sum_{j=1}^{N_r} |u(\boldsymbol{m}; p_j)|,$$

where the notation $u(\boldsymbol{m}; p)$ means the wavefield $u(\boldsymbol{m})$ evaluated at the sensor position p. By Assumptions 2.4.10, Point 5, if the sensors are not located at the source positions, we can write

$$||\boldsymbol{\varepsilon}(\boldsymbol{m}, \mathcal{P})||_1 \leq ||\mathbf{d}||_1 + N_r C_2(\omega). \tag{2.4.44}$$

■

Combining Theorem 2.4.12 and Lemma 2.4.14, we obtain the following.

Corollary 2.4.15. ***Upper Bound on*** $H^{(2)}$*:* *If Assumption 2.4.10 holds, then*

$$\left| \frac{\left\langle H^{(2)}(\boldsymbol{m}, \mathcal{P})\widetilde{\boldsymbol{m}}, \widetilde{\boldsymbol{m}} \right\rangle}{\langle \widetilde{\boldsymbol{m}}, \widetilde{\boldsymbol{m}} \rangle} \right| \leq C(\omega) \left(||\boldsymbol{d}||_1 + N_r C_2(\omega) \right). \tag{2.4.45}$$

We introduce the following notation for the constant in the bound in Corollary 2.4.15 as

$$\mathsf{H}^2 := C(\omega) \left(||\mathbf{d}||_1 + N_r C_2(\omega) \right). \tag{2.4.46}$$

We now turn our attention to bounds on $H^{(1)}$.

Theorem 2.4.16. ***Upper and Lower Bounds on*** $H^{(1)}$*:* *If Assumption 2.4.10 holds, then for all* $\boldsymbol{m} \in \mathcal{M}$ *and all* $\widetilde{\boldsymbol{m}} \in \mathbb{R}^M$*,*

$$0 \leq \frac{\left\langle H^{(1)}(\boldsymbol{m}, \mathcal{P})\widetilde{\boldsymbol{m}}, \widetilde{\boldsymbol{m}} \right\rangle}{\langle \widetilde{\boldsymbol{m}}, \widetilde{\boldsymbol{m}} \rangle} \leq \omega^4 N_r C_3(\omega) C_1^3(\omega).$$

Proof. By definitions (2.2.18) and (2.2.17)

$$\mathcal{R}(\mathcal{P})^*(\mathcal{R}(\mathcal{P})v(\boldsymbol{m},\widetilde{\boldsymbol{m}})) = \sum_{j=1}^{N_r} \delta_{p_j} v(\boldsymbol{m},\widetilde{\boldsymbol{m}};p_j). \tag{2.4.47}$$

Using (2.4.47) in the definition (2.4.25) of z_1, we get

$$\begin{aligned} \|z_1(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P})\|_{L^2(\Omega)} &= \|\mathscr{S}_{\boldsymbol{m}}^*(\mathcal{R}(\mathcal{P})^*\mathcal{R}(\mathcal{P})v(\boldsymbol{m},\widetilde{\boldsymbol{m}}))\|_{L^2(\Omega)}, \\ &= \left\| \mathscr{S}_{\boldsymbol{m}}^* \left(\sum_{j=1}^{N_r} \delta_{p_j} v(\boldsymbol{m},\widetilde{\boldsymbol{m}};p_j) \right) \right\|_{L^2(\Omega)}. \end{aligned} \tag{2.4.48}$$

By the linearity of $\mathscr{S}_{\boldsymbol{m}}^*$, and Assumption 2.4.10 Point 4,

$$\begin{aligned} \|z_1(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P})\|_{L^2(\Omega)} &\leq \sum_{j=1}^{N_r} |v(\boldsymbol{m},\widetilde{\boldsymbol{m}};p_j)| \|\mathscr{S}_{\boldsymbol{m}}^*(\delta_{p_j})\|_{L^2(\Omega)} \\ &\leq C_1(\omega)||v(\boldsymbol{m},\widetilde{\boldsymbol{m}};\mathcal{P})||_1. \end{aligned}$$

By Assumptions 2.4.10 Point 6, and the definition (2.4.18) of v,

$$\begin{aligned} |v(\boldsymbol{m},\widetilde{\boldsymbol{m}};p_j)| &\leq C_3(\omega)\omega^2||\widetilde{m}u(\boldsymbol{m})||_{L^2(\Omega)} \\ &\leq C_3(\omega)\omega^2||\widetilde{m}||_{L^\infty(\Omega)}||u(\boldsymbol{m})||_{L^2(\Omega)} \\ &\leq C_3(\omega)C_1(\omega)\omega^2||\widetilde{m}||_{L^\infty(\Omega)} \end{aligned}$$

and so

$$||v(\boldsymbol{m},\widetilde{\boldsymbol{m}};\mathcal{P})||_1 = \sum_{j=1}^{N_r} |v(\boldsymbol{m},\widetilde{\boldsymbol{m}};p_j)| \leq N_r C_3(\omega)C_1(\omega)\omega^2||\widetilde{m}||_{L^\infty(\Omega)}. \tag{2.4.49}$$

By (2.4.27), (2.4.48) and (2.4.49),

$$\begin{aligned} \left\langle H^{(1)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\right\rangle &\leq \omega^2\|\widetilde{m}\|_{L^\infty(\Omega)}\|u(\boldsymbol{m})\|_{L^2(\Omega)}\|z_1(\boldsymbol{m},\widetilde{\boldsymbol{m}},\mathcal{P})\|_{L^2(\Omega)} \\ &\leq \omega^2\|\widetilde{m}\|_{L^\infty(\Omega)}C_1^2(\omega)||v(\boldsymbol{m},\widetilde{\boldsymbol{m}};\mathcal{P})||_1. \\ &\leq \omega^2 C_1^2(\omega)||\widetilde{m}||_{L^\infty}\left(\omega^2 N_r C_3(\omega)C_1(\omega)\|\widetilde{m}\|_{L^\infty(\Omega)}\right) \\ &= \omega^4 N_r C_3(\omega)C_1^3(\omega)\|\widetilde{m}\|_{L^\infty(\Omega)}^2 \end{aligned}$$

Using that $||\widetilde{m}||_{L^\infty} \leq ||\widetilde{\boldsymbol{m}}||_\infty \leq ||\widetilde{\boldsymbol{m}}||_2$, we have the upper bound

$$\frac{\left\langle H^{(1)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\right\rangle}{\langle\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\rangle} \leq \omega^4 N_r C_3(\omega)C_1^3(\omega). \tag{2.4.50}$$

The lower bound is found by combining the definitions (2.4.3) and (2.4.9), so that

$$\begin{aligned}\left\langle H^{(1)}(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\right\rangle &= \Re\left\langle \mathcal{R}(\mathcal{P})v(\boldsymbol{m},\widetilde{\boldsymbol{m}}),\mathcal{R}(\mathcal{P})v(\boldsymbol{m},\widetilde{\boldsymbol{m}})\right\rangle \\ &= \Re\left(||\mathcal{R}(\mathcal{P})v(\boldsymbol{m},\widetilde{\boldsymbol{m}})||_2^2\right) \\ &= ||\mathcal{R}(\mathcal{P})v(\boldsymbol{m},\widetilde{\boldsymbol{m}})||_2^2 \qquad \geq 0. \end{aligned} \tag{2.4.51}$$

Combining (2.4.50) and (2.4.51) gives the result. ∎

We introduce the following notation for the constant in the upper bound in Theorem 2.4.16

$$\mathsf{H}^1 := \omega^4 N_r C_3(\omega) C_1^3(\omega). \tag{2.4.52}$$

Upper and lower bounds for the Tikhonov regularisation term in (2.4.38) are now discussed. Denoting the maximum eigenvalue of D^TD as C_D, and noting that D^TD is positive semi-definite, we can write the following bounds,

$$0 \leq \frac{\left\langle D^TD\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\right\rangle}{\langle\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\rangle} \leq C_D \tag{2.4.53}$$

We now combine the bounds we have found so far to find upper and lower bounds on the regularised Hessian.

Corollary 2.4.17. ***Upper and Lower Bounds on*** H*:* *The range of eigenvalues of the regularised Hessian* (2.4.38) *is*

$$\mu - \mathsf{H}^2 \leq \frac{\langle H(\boldsymbol{m},\mathcal{P})\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\rangle}{\langle\widetilde{\boldsymbol{m}},\widetilde{\boldsymbol{m}}\rangle} \leq \mu + \mathsf{H}^2 + \mathsf{H}^1 + \alpha C_D \tag{2.4.54}$$

where H^1 *and* H^2 *are defined by* (2.4.52) *and* (2.4.46) *respectively.*

Proof. By Proposition 2.4.11, Theorems 2.4.12, 2.4.16, and (2.4.53), the result (2.4.54) holds. ∎

Theorem 2.4.18. ***Conditions for a Positive Definite*** H*:* *If Assumptions 2.4.10 hold, then if the regularisation parameter* μ *is chosen to be*

$$\mu = \mathsf{H}^2 + \tau$$

for some constant $\tau > 0$*, then the Hessian* $H(\boldsymbol{m},\mathcal{P})$ *given in* (2.4.38) *is positive definite for all* $\boldsymbol{m} \in \mathcal{M}$*.*

Proof. By Corollary 2.4.17, by setting μ equal to $\mathsf{H}^2 + B$, the eigenvalue range of H is

$$\tau \leq \frac{\langle H(\boldsymbol{m}, \mathcal{P})\widetilde{\boldsymbol{m}}, \widetilde{\boldsymbol{m}}\rangle}{\langle \widetilde{\boldsymbol{m}}, \widetilde{\boldsymbol{m}}\rangle} \leq 2\mathsf{H}^2 + \mathsf{H}^1 + \alpha C_D + \tau.$$

Therefore the smallest eigenvalue of H is bounded below by a positive constant τ, making the Hessian positive definite. This choice of μ is independent of $\boldsymbol{m}$ so, by Corollary 2.4.15, H is positive definite for all $\boldsymbol{m} \in \mathcal{M}$. ∎

Remark 2.4.19. ***Size of μ in Practice:*** *Note that the choice of μ in Theorem 2.4.18 is a lower bound for μ that will definitely ensure the positive definiteness of the Hessian, and it is possible that a smaller μ will still result in a positive definite Hessian. In practice, it is expected that the Tikhonov regularisation term and $H^{(1)}$ term will help the positive-definiteness of the Hessian, since the lower-bound on the smallest eigenvalue of a sum of matrices is only reached in the case where the matrices in the sum share eigenvectors (by Weyl's theorem [94, Theorem 4.3.1]). It is observed throughout the later computations in this thesis that a positive-definite Hessian is obtained even for small values of μ.*

Corollary 2.4.20. *When the regularisation parameter μ is chosen to be*

$$\mu = \mathsf{H}^2 + \tau$$

for some constant $\tau > 0$, then a minimiser of the FWI problem exists and is unique.

Proof. By Theorem 2.4.18, this choice of μ makes the FWI Hessian positive definite, where the smallest eigenvalue of the Hessian is bounded below by a positive constant τ, for all models $\boldsymbol{m} \in \mathcal{M}$. By the second-order characterisation of strong convexity (Appendix H, Lemma H-5), the objective function $\phi(\boldsymbol{m}, \mathcal{P})$ (2.4.37) is strongly convex. By Theorem H-6 in Appendix H, a minimum of ϕ exists and is unique. ∎

We note that all theorems and corollaries in this Section 2.4.4 have been proved assuming one source and one frequency. To extend Corollary 2.4.20 to the multiple sources and frequencies case, then H^2 in (2.4.46) should be summed over all sources and frequencies.

Remark 2.4.21. ***Disadvantage of Convex Regularisation Term:*** *This approach of adding a convex regularisation term to the FWI objective function can be a useful trick to modify the FWI problem so that it has a unique solution. However, it is important to note that if the FWI problem is very non-convex, forcing it to be strongly convex with a regularisation term can mean that interesting information coming from the data misfit term is lost in the solution to the problem.*

2.5 FWI Algorithm

This section provides details on the FWI algorithm in the frequency domain. To describe the algorithm, we first motivate the frequency continuation approach, which is used to improve FWI performance.

2.5.1 Frequency Continuation

One of the main difficulties that arises from the non-linearity of FWI is the presence of numerous local minima. The presence of local minima means that, unless the starting guess of the subsurface model is close to the global minimum, the optimisation method may not converge to it. This issue led to the development of some hierarchical 'multiscale strategies' to mitigate this local minima problem. Here, the term 'multiscale strategies' refer to methods that successively process data subsets of increasing resolution. This multiscale idea was originally proposed for FWI in the time domain by Bunks et al. [38], and later formulated and implemented in the frequency domain, for example in [150] and [149].

Physically, during acquisition, lower frequency components of the recorded data will generally be due to the wavefield scattering from larger structures in the subsurface, whereas higher frequency components will contain the detail of smaller features. Therefore, the data in the FWI objective function contains information at various scales of the subsurface. The FWI objective function has fewer local minima for lower frequency data [38]. Higher frequency components are likely to have been scattered many times, and so there are more local minima when FWI is performed for higher frequency data. Mathematically, lower frequencies increase the radius of convergence for the FWI problem (see [69, Chapter 5]), whereas higher frequencies are necessary to improve the resolution of the FWI reconstruction (this fact is motivated by the stability result referenced in Section 2.2.5 - see [69, Chapter 3] for more details).

These properties led to the suggestion in [38] to decompose the problem by scale, successively performing the inversion from long scales (low frequency) to short scales (high frequency), and hence reconstructing large scale features before resolving the finer details. This approach occurs more naturally in the frequency domain, by performing successive inversions of increasing frequencies. This approach is termed *frequency continuation*, or frequency progression. This frequency continuation strategy involves initially performing FWI for low frequency data only, in order to obtain a low resolution model. This model is then used as the starting model for the next inversion, using higher frequency data. Following that, FWI is performed for higher and higher frequency data to obtain more resolution in the model. Multiple frequencies (i.e., a 'frequency group') are typically inverted at each stage of the method.

The algorithm for Frequency Continuation is given as Algorithm 2.5.1 below. It involves splitting frequencies into groups ordered from lowest to highest. FWI is performed for a chosen starting guess $\boldsymbol{m}_0$ for the first group g_1, and the result $\boldsymbol{m}^{FWI}$ is then used as a starting guess for the next round of FWI with frequency group g_2. This process continues until all frequency groups have been looped through. In the

algorithm, FWI($g_k, \boldsymbol{m}_0$) means performing FWI for a group of frequencies g_k, with the initial guess as $\boldsymbol{m}_0$.

Algorithm 2.5.1 Frequency Continuation

1: *Inputs:* Initial model $\boldsymbol{m}_0$, Frequencies from set $\mathcal{W}$ given in increasing order $\omega_1 < \omega_2 < ... < \omega_{N_\omega}$
2: Group frequencies into N_f groups $\{g_1, g_2, ... , g_{N_f}\}$
3: **for** $k = 1$ **to** N_f **do**
4: $\quad \boldsymbol{m}^{FWI} \leftarrow$ FWI(g_k, $\boldsymbol{m}_0$)
5: $\quad \boldsymbol{m}_0 \leftarrow \boldsymbol{m}^{FWI}$
6: **end for**
7: *Output:* Optimal model $\boldsymbol{m}^{FWI}$

The computational cost of FWI depends on the number of frequencies used (as demonstrated in the gradient and Hessian derivations in Sections 2.3 and 2.4). Sirgue and Pratt [166] show that a good quality FWI image can be obtained using a very limited number of frequencies, which is a significant advantage for computational efficiency. In [166], a strategy for selecting these frequencies, for a reflection setup, is outlined, where the idea is that the larger the maximum offset (horizontal distance from source to receiver) is, the fewer frequencies are needed. They show that frequency domain FWI with reflection data, using only a few properly selected frequencies, can produce a result that is comparable to full-time domain FWI.

2.5.2 Full Algorithm

Here we detail a possible FWI algorithm, where we assume we are given data $\mathbf{d}$ that comes from a known acquisition setup. The algorithm is written with discretised variables, to make it relevant for implementation. We present the case where we do the inversion for a frequency group of size N_g, with frequencies $\{\omega_1, ..., \omega_{N_g}\}$, which are a subset of the set of frequencies $\mathcal{W}$. The FWI algorithm would be then called within the frequency continuation algorithm (Algorithm 2.5.1). We break the FWI algorithm into two parts, the optimisation algorithm (Algorithm 2.5.2), and the evaluation of the objective function and its gradient (Algorithm 2.5.3).

We write the FWI optimisation algorithm (Algorithm 2.5.2) without specifying the details of the descent direction or line search computation. We write the algorithm generally to indicate that any gradient-based optimisation method can be chosen to find the descent direction and any line search method can be used to find the step size. For example, in the implementation in Section 2.6, L-BFGS (Algorithms F-1 and F-2) and Wolfe Line Search (Algorithm F-3) are used to find the descent direction and step size respectively. In this case, computing the descent direction requires the current and past values of the FWI gradient $\nabla\phi$, and computing the line search direction involves calling the function for the computation of $\nabla\phi$ (i.e., Algorithm 2.5.3) within it.

A schematic is included to demonstrate the general workflow of FWI in Figure 2.5.1. Note in the algorithms and schematic, regularisation is not included for brevity.

Regularisation may be included by adding the appropriate terms to ϕ (as in 2.4.37). We note that in these algorithms, and all other algorithms written in this thesis, when another algorithm is called within the original algorithm, we indicate the inputs of this inner algorithm by writing them in brackets.

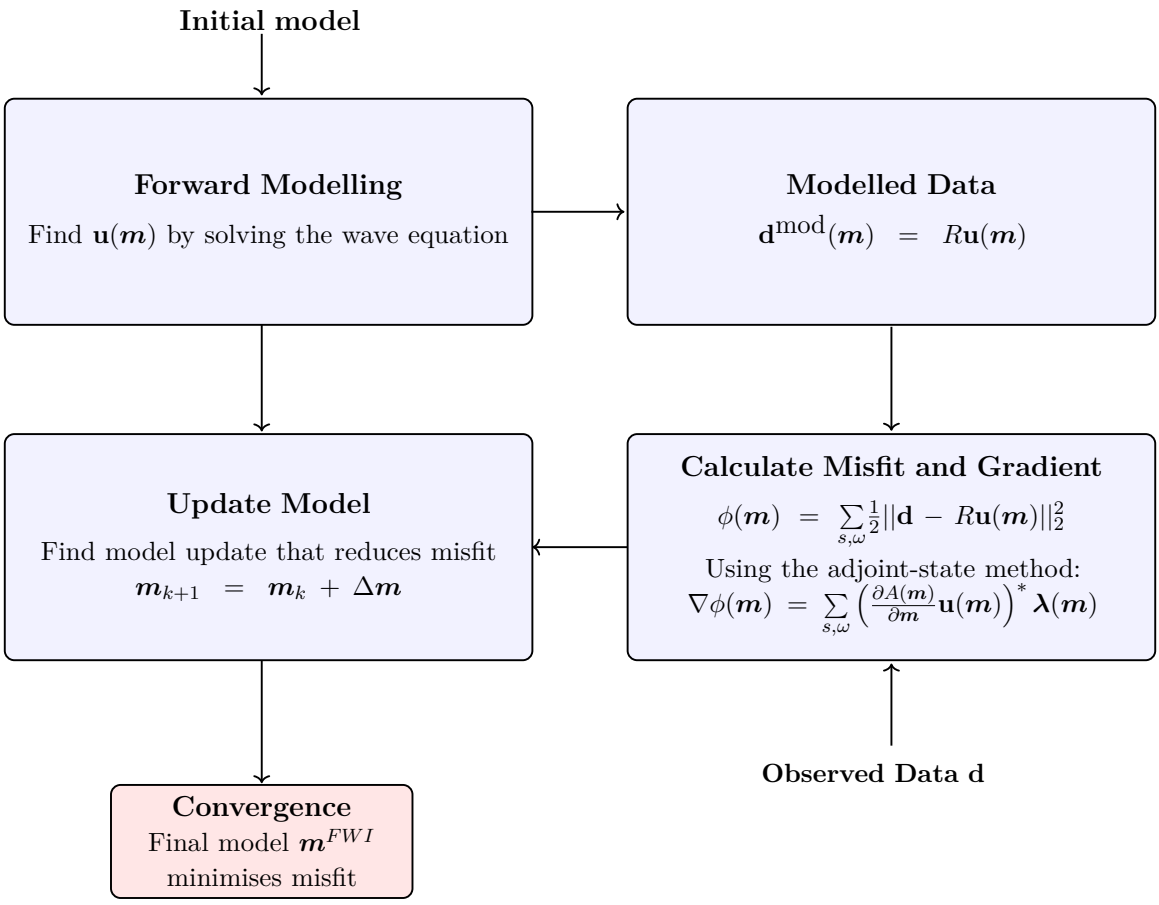


Figure 2.5.1: *Schematic of main FWI steps*

Algorithm 2.5.2 FWI Optimisation Algorithm

Inputs: Initial guess model $\boldsymbol{m}_0$, data $\mathbf{d}$, source positions $\mathcal{S}$, frequency group g, sensor positions $\boldsymbol{p}$, convergence tolerance tol, maximum iterations $k_{\max}$
Compute $\nabla\phi(\boldsymbol{m}_0)$ with `Algorithm 2.5.3`$(\boldsymbol{m}_0, \mathbf{d}, \mathcal{S}, g, \boldsymbol{p})$
$k = 0$
while $||\nabla\phi(\boldsymbol{m}_k)|| > tol$ and $k > k_{\max}$ **do**
    $\mathbf{d}_k \leftarrow$ Descent Direction
    $\alpha_k \leftarrow$ Line Search
    $\boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \alpha_k \mathbf{d}_k$
    Compute $\nabla\phi(\boldsymbol{m}_{k+1})$ with `Algorithm 2.5.3`$(\boldsymbol{m}_{k+1}, \mathbf{d}, \mathcal{S}, g, \boldsymbol{p})$
    $k = k + 1$
end while
Output: Optimal model $\boldsymbol{m}^{FWI}$

Algorithm 2.5.3 FWI Objective Function and Gradient

Inputs: Frequency group g of size N_g, Source positions $\mathcal{S}$, sensor positions $\boldsymbol{p}$, Observed data $\mathbf{d}$, discretised model $\boldsymbol{m}$
Initialise $\phi = 0$, $\nabla\phi = \mathbf{0}$
Compute restriction operator $R(\boldsymbol{p})$
for $s \in \mathcal{S}$ **do**
    for $k \in \{1, .., N_g\}$ **do**
        Assemble source vector $\mathbf{f}(s, \omega_k)$
        Assemble system matrix $A(\boldsymbol{m}, \omega_k)$
        Compute $\mathbf{u}(\boldsymbol{m}, s, \omega_k)$ from (2.2.14)
        Compute modelled data $\mathbf{d}^{mod}(\boldsymbol{m}, \boldsymbol{p}, s, \omega_k) = R(\boldsymbol{p})\mathbf{u}(\boldsymbol{m}, s, \omega_k)$
        Evaluate misfit $\boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{p}, s, \omega_k) = \mathbf{d} - \mathbf{d}^{mod}(\boldsymbol{m}, \boldsymbol{p}, s, \omega_k)$
        Compute adjoint wavefield $\boldsymbol{\lambda}$ form (2.3.12)
        Update $\phi = \phi + \frac{1}{2}||\boldsymbol{\varepsilon}||_2^2$
        for $j \in \{1, ..., M\}$ **do**
            Update $\nabla\phi_j = \nabla\phi_j + \left(\frac{\partial A}{\partial m_j}\mathbf{u}\right)^* \boldsymbol{\lambda}$
        end for
    end for
end for
Outputs: ϕ, $\nabla\phi$

2.6 FWI Reconstructions

In this section the FWI algorithm is used to create images of artificial subsurfaces. Our FWI experiments are for acoustic media with constant density. Therefore, the wave propagation is described by the Helmholtz equation, and the inverse problem aims to recover/reconstruct the wavespeed/velocity. The objective function being minimised is of the form (2.4.37), with regularisation parameters chosen through experimentation, and the iterative minimisation algorithm used to reconstruct the model is L-BFGS (Algorithms F-1 and F-2) and Weak Wolfe Line Search (Algorithm F-3). We note that the finite difference discretisation of the Helmholtz equation used here comes from [180], and that the rest of the FWI code is original.

Remark 2.6.1. ***Synthetic 'Observed' Data*** *In the case where data does not come from acquisition, an extra step must be included to compute synthetic data to use for* $\boldsymbol{d}$*. The synthetic data is computed from a forward modelling code using a given model, and restricting the solution to the sensor positions. The forward modelling for the purpose of creating synthetic data should be done using a different grid as that used to compute the modelled data to avoid an inverse crime (see Definition 2.4.5).*

All 'observed' data in these experiments are synthetic.

Experiment 1

The first experiment involves a simple velocity model with a region of higher wavespeed in the centre that smoothly decreases outwards. We consider a 2500 m $\times$ 2500 m domain, with maximum velocity 2100 ms^{-1} in the centre. The model is discretised into a 101 $\times$ 101 grid, with a spacing of 25 m. This discretisation results in 10201 model parameters to be recovered.

Acquisition is simulated using a crosswell setup, making this a transmission experiment. Five sources are located in a well on the left of the domain, and five sensors are located in a parallel well on the opposite side. The acquisition setup is illustrated on the ground truth model in Figure 2.6.1.

We make an initial guess at the velocity model, shown in Figure 2.6.2 (a), with the aim of recovering the true model in Figure 2.6.1. We choose to do FWI for one frequency only due to the simplicity of the model. Both a Tikhonov and convex regularisation term are included in the objective function, with parameters $\alpha = 0.5$ and $\mu = 10^{-7}$ respectively. The optimisation algorithm is iterated until we reach the stopping condition $||\nabla\phi||_2 \leq 10^{-6}$, i.e., the norm of the gradient must be small enough. Figure 2.6.2 (b) shows the model that we recover. The colour scale is the same as for Figure 2.6.1.

Figure 2.6.3 (a) shows the evolution of the objective function with iterations. There is initially a large reduction in the objective function, after which the decrease of the cost function slows down. This happens here after about 10 iterations. A stopping condition can be included in FWI to stop iterations after the value of the objective function stagnates, however we have continued to iterate to demonstrate the behaviour of the gradient (Figure 2.6.3 (b)). While the reduction in the objective function has

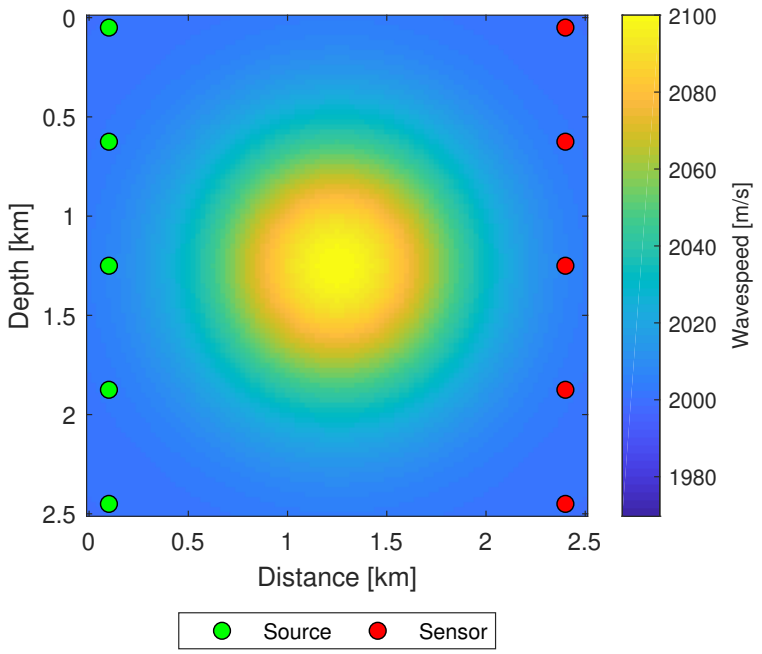
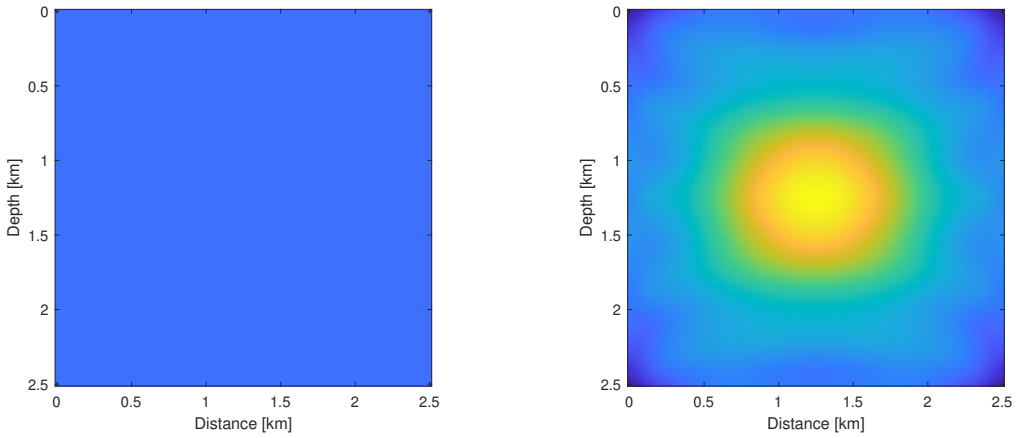


Figure 2.6.1: *Ground truth velocity model to be reconstructed with FWI. Sources and sensors are placed in parallel wells on opposite side of the domain, meaning that transmitted data is recorded at the sensors, i.e., the sensors measure the wavefield which has been transmitted through the domain.*

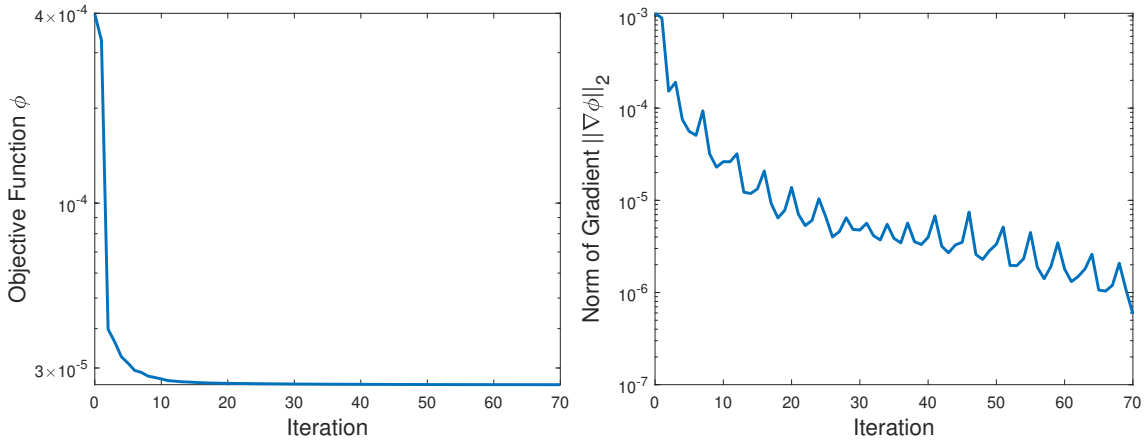


(a) Initial Wavespeed Guess

(b) FWI Solution

Figure 2.6.2: *Initial and Final velocity models in the FWI reconstruction of Figure 2.6.1.*

stalled, the gradient continues to decrease. (This point will become important in the parameter optimisation problem and is discussed more in Section 5.3.3). Note that the small increases in the graident norm every few iterations is comented on in [135].



(a) Objective Function ϕ (b) Norm of Gradient $||\nabla\phi||_2$

Figure 2.6.3: *Evolution of the FWI objective function and its gradient on a log scale.*

Experiment 2

This experiment considers a more complicated two-dimensional velocity model, the acoustic *Marmousi* model. The Marmousi model is a semi-real geophysical model, synthetically designed by the Institut Français du Pétrole (IFP) in 1988. It has since been thoroughly studied and is one of the most popular models used in geophysical applications to test inversion procedures. The model consists of layered structures, with strong horizontal and vertical velocity changes.

We consider the Marmousi model, which has been smoothed horizontally and vertically to obtain a *smooth Marmousi model*. This smoothing is performed by applying a Gaussian filter to the original Marmousi model. When smoothed, less layers can be identified in the model. This smooth Marmousi model is shown in Figure 2.6.4. We note that various 'smoothed' versions of the Marmousi model have been used in testing geophysical algorithms, for example in [160], [174], [191], and [187].

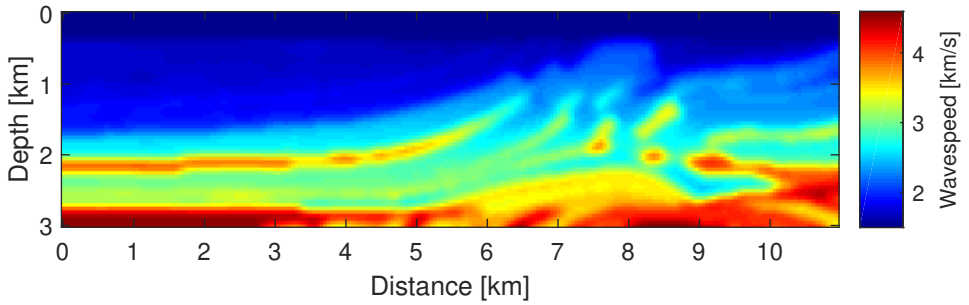


Figure 2.6.4: *Smooth Marmousi model. We aim to reconstruct this ground truth model with FWI.*

The domain of the model is 10.95 km $\times$ 3 km, and the wavespeed varies from 1.5 kms^{-1} to 4.59 kms^{-1}. The domain is discretised into a 220 $\times$ 61 grid, with a spacing of 50 m in each direction. The model therefore has 13420 parameters.

The seismic acquisition is designed with sources and sensors in the near surface area. This is a *reflection* acquisition setup. We use 55 sources, equally spaced by 200 m along a line at a depth of 250 m, and 110 sensors, equally spaced by 100 m along a line at a depth of 100 m. Synthetic data is generated using a different grid to that used in the FWI algorithm to recover the model parameters (specifically a 121×439 grid).

From this synthetic data, we aim at reconstructing Figure 2.6.4 without knowing any information about the subsurface structures. The initial guess at the velocity is shown in Figure 2.6.5. The initial guess is simply a one-dimensional linear variation of the velocity, with increasing value with depth.

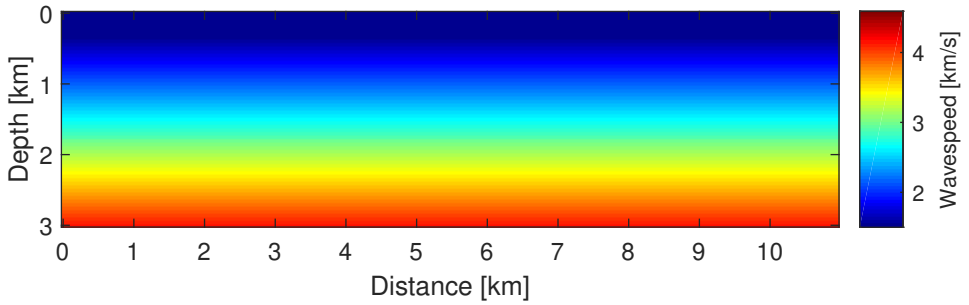


Figure 2.6.5: *Initial wavespeed for the iterative optimisation algorithm. The model consists of a one-dimensional linear variation of velocity with depth and has no information on any of the true structures.*

The FWI objective function being minimised contains a Tikhonov and convex regularisation parameter, with parameters $\alpha = 1$ and $\mu = 10^{-6}$. We perform frequency continuation (Algorithm 2.5.1) with 3 groups, where the frequencies are chosen sparsely according to the selection strategy outlined in [166]. The optimisation algorithm is iterated until the norm of the gradient of the objective function reaches the specified tolerance (10^{-6}) for each frequency group. The resulting reconstructed wavespeed is shown in Figure 2.6.6. The 'macro' structure of the layers appears to be relatively accurate, and to the eye looks similar to the ground truth. However, some of the deeper layers are not as well-recovered and have a blob-like appearance.

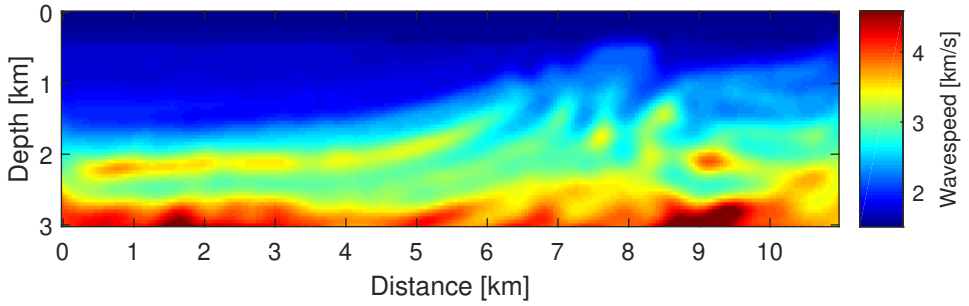
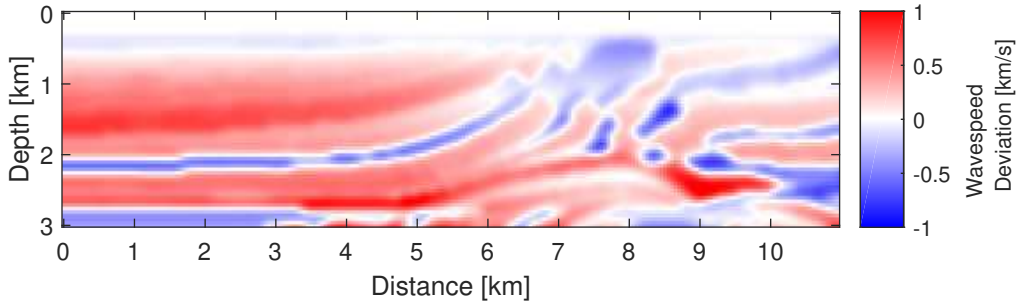


Figure 2.6.6: *Final reconstruction of the smooth Marmousi model using FWI with frequency continuation.*

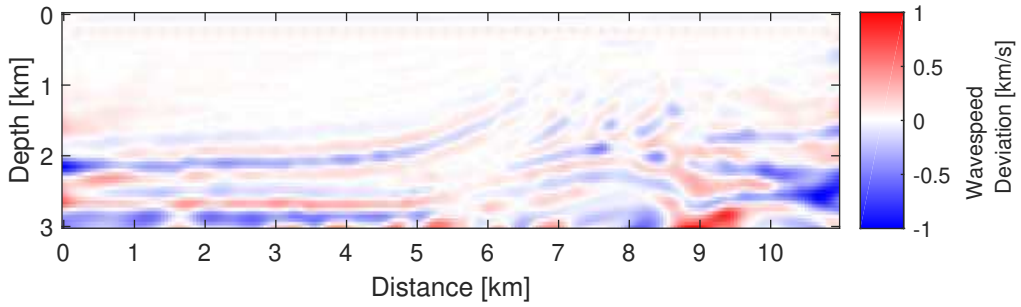
We now analyse the quality of the FWI image of the subsurface more carefully. To highlight the differences between the ground truth and the corresponding reconstruction, we look at the differences in their wavespeed values at each grid point location. These differences in wavespeed are called the *deviation* from the true subsurface, or the absolute error, which we will define as

$$Absolute\ Error\ =\ Reconstructed\ Value\ -\ True\ Value. \tag{2.6.1}$$

To show the improvement in the FWI image from the starting image, we also look at the absolute error in the starting guess. The absolute error is shown in Figure 2.6.7. Figure (a) shows this error for the initial guess (i.e., wavespeeds in Figure 2.6.5 minus wavespeeds in Figure 2.6.4). Figure (b) shows the error for the FWI reconstructed image (i.e., wavespeeds in Figure 2.6.6 minus wavespeeds in Figure 2.6.4).



(a) Deviation in **initial** wavespeed from the ground truth



(b) Deviation in the **FWI reconstruction** from the ground truth

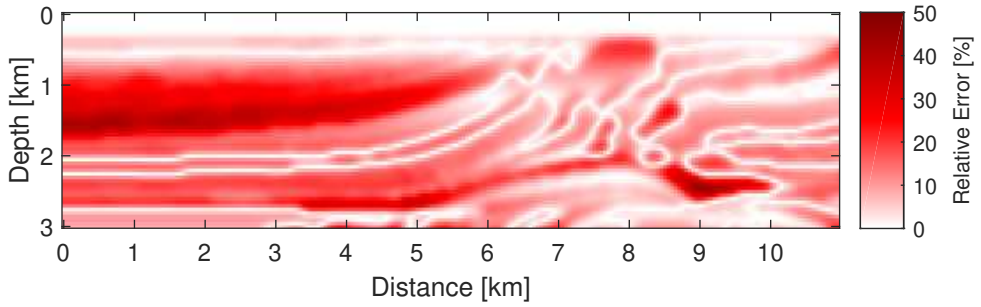
Figure 2.6.7: *The deviation of the starting and final wavespeed from the ground truth.*

Clearly, the initial guess contains none of the structures present in the ground truth. FWI recovers the layered structure well, particularly in shallow regions, where the deviation is close to zero (seen in Figure 2.6.7 (b) as white). At a depth of 2km and lower however, the errors are intensified. The deepest parts of the model have the sparsest data coverage due to the reflection setup, and so poor recovery is expected (noted in [132]).

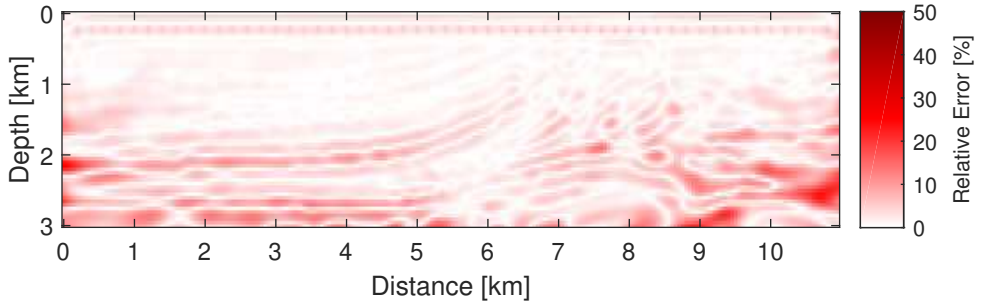
The relative percentage error is the absolute error as a percentage of the true values. We plot the absolute value of this in Figure 2.6.8. Specifically, we are plotting

$$Relative\ Error\ =\ \left|\frac{Reconstructed\ Value\ -\ True\ Value}{True\ Value}\right| \times 100. \tag{2.6.2}$$

We also plot the relative error for the starting guess for comparison.



(a) Absolute value of the relative percentage error in the **initial** guess



(b) Absolute value of the relative percentage error in the **FWI reconstruction**

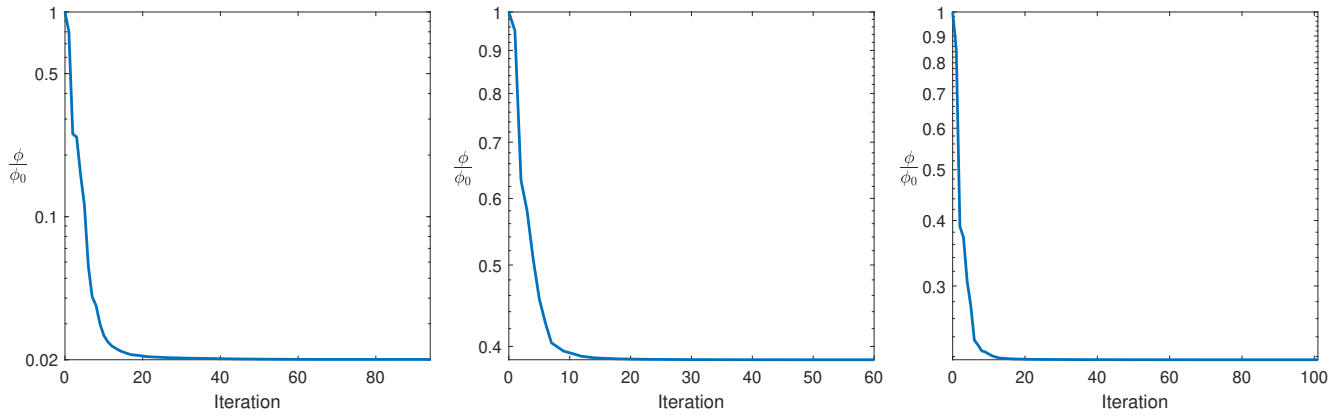
Figure 2.6.8: *The relative error in the starting and final wavespeeds.*

Initially, in Figure 2.6.8 (a), the relative errors were as high as 44%. The median relative error is 8.27%. After FWI, the percentage errors were significantly reduced at all depths, as shown in Figure 2.6.8 (b). The median error has been reduced to 1.69%. There are still some regions with larger errors, particularly at deeper depths, with the maximum relative error (25.49%) occurring at a depth of 2.15 km on the left of the domain, which corresponds to the edge of a layer of high wavespeed. Other areas with large errors (indicated by deep red) in the reconstruction are regions in the bottom right. In general, the largest errors are located at the edges of the model and at depth. This fact is also noted in [124]. We also observe errors near the source and sensor locations that are larger than errors in the surrounding areas. Artefacts resulting from the acquisition geometry are termed an 'acquisition footprint' [50]. These artefacts can be reduced or removed by a denser acquisition layout, increased regularisation or postprocessing techniques [82].

Other measures of error can be used to summarise the overall error. One example is squared model error. Initial squared model error is defined by $||\boldsymbol{m}^{true} - \boldsymbol{m}_0||_2^2$, which is 30.93 here, and final squared model error, defined by $||\boldsymbol{m}^{true} - \boldsymbol{m}^{FWI}||_2^2$, which is 1.74 here (where we have rounded values to two decimel places). Other measures of the image quality exist in geophysics, for example the structural similarity (SSIM), which is a measure of the similarity between images (for example see [57]).

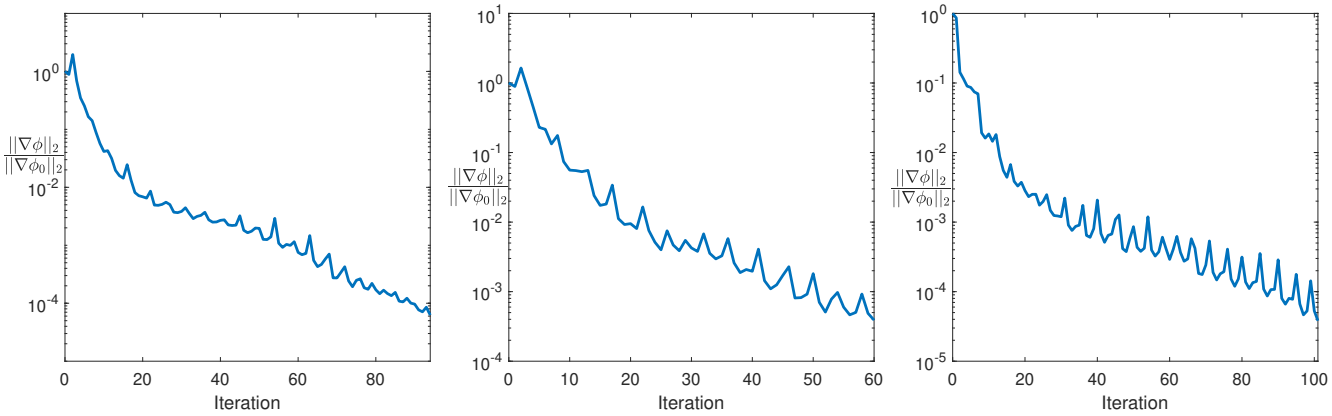
The evolution of the objective function for each frequency group is shown in Figure

2.6.9, and the evolution of the norm of the gradient of the objective function is shown in Figure 2.6.10. We have scaled these values by their initial value so that the results for each frequency group can be compared. We see similar behaviour to Experiment 1, where the decrease in the objective function stalls early, but the decrease in the gradient continues.



(a) Group 1 (b) Group 2 (c) Group 3

Figure 2.6.9: *Evolution of the FWI objective function on a log scale, shown for each frequency group (low to high). The objective function ϕ is scaled by its initial value ϕ_0.*

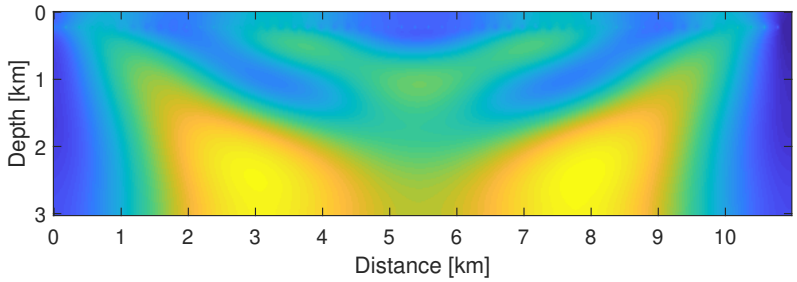


(a) Group 1 (b) Group 2 (c) Group 3

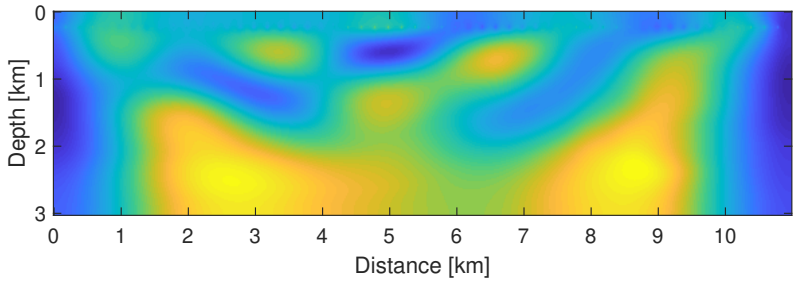
Figure 2.6.10: *Evolution of the norm of the gradient of the FWI objective function on a log scale, shown for each frequency group (low to high). The norm of the gradient $||\nabla\phi||_2$ is scaled by its initial value $||\nabla\phi_0||_2$.*

Figures 2.6.11 and 2.6.12 show the absolute value of the wavefield $|\mathbf{u}|$, produced by all 55 sources, computed by solving the Helmholtz equation. Figure 2.6.11 is the Helmholtz solution at the lowest frequency used in the FWI frequency continuation algorithm, and Figure 2.6.12, shows the Helmholtz solution at the highest frequency used. In each plot, subplot (a) shows the wavefield due to the starting guess model, and

subplot (b) shows the wavefield for the true model. These plots demonstrate how the wave interacts with the different layers of the Marmousi model, compared to a smooth model. The point sources are faintly visible along 0.25 km depth. We note that these figures involve a superposition of wavefields from all sources for visualisation purposes, but during FWI the wavefield from each source is processed individually.

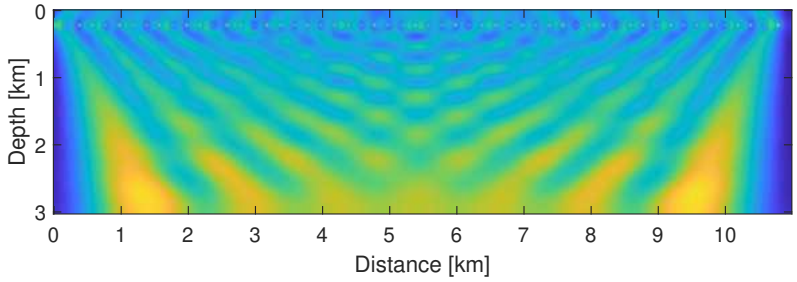


(a) Wavefield for starting model (Figure 2.6.5)

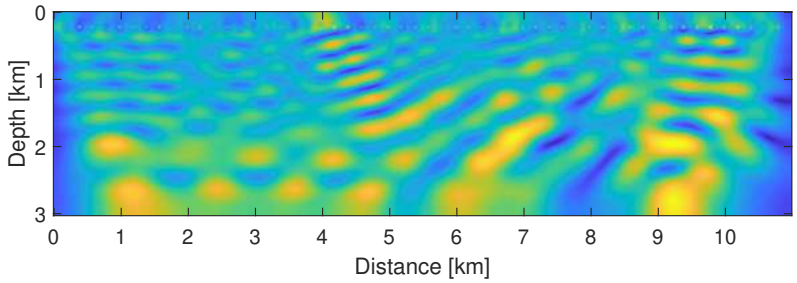


(b) Wavefield for ground truth (Figure 2.6.4)

Figure 2.6.11: *Wavefield for lowest frequency used in the FWI frequency continuation.*



(a) Wavefield for starting model (Figure 2.6.5)



(b) Wavefield for ground truth (Figure 2.6.4)

Figure 2.6.12: *Wavefield for highest frequency used in the FWI frequency continuation.*

Chapter 3

Parameter Optimisation

Chapter Summary: This chapter focuses on parameter optimisation in seismic imaging, starting with the sensor placement optimisation problem and then incorporating the optimisation of a FWI regularisation parameter. We review the motivation (§3.1.1) and current approaches (§3.1.2) to the sensor placement problem, before introducing the novel approach of this PhD thesis to sensor optimisation (§3.2). We formulate the sensor placement optimisation problem in the framework of bilevel learning (§3.3), derive a gradient-based optimisation method for the solution of the bilevel problem (§3.4.2), derive the cost of this solution method in terms of the number of PDE solves (Theorem 3.4.10) and investigate the smoothness of the FWI solution with respect to sensor position (§3.4.3). The same analysis is completed for the optimisation of the FWI Tikhonov regularisation parameter (§3.5). We also provide some simple examples of sensor placement and regularisation parameter optimisation (§3.6). In this chapter, §3.3 onwards is novel work.

3.1 Introduction to Experimental Design

3.1.1 Motivation for Seismic Survey Design

Seismic surveys, as discussed in Sections 1.3 and 1.4, are a method of gathering information about the geological properties of the Earth's subsurface. The information acquired is used to produce maps of the underground, allowing the identification of areas of interest, for example areas where gas or oil deposits may be found. Seismic survey design involves prior planning of the acquisition process such that the objectives of the survey will be met. Survey design is a crucial step, as it dictates the quality of the information collected during acquisition, and ultimately the quality of the subsurface images produced. If the design is poor and the data collected is inadequate, or data that would have been vital in reconstructing geological structures is missing, no amount of subsequent data processing can make up for this [123]. It is therefore essential that acquisition procedures have been designed to maximise the desired information.

A simple, and occasionally used, approach to the problem of insufficient and inadequate data is to collect as much data as possible [123]. This approach potentially

results in a large amount of redundant data. Cost is a major consideration in seismic surveys and so it is critical to ensure that industry achieves the best return on their investment into the acquisition. This suggests that the overall goal in seismic survey design is to plan cost-effective procedures for acquiring optimal data (i.e., data that contains the most useful information in terms of resolving the specific subsurface features or parameters of interest). Good quality survey design is therefore important in justifying the cost of the seismic experiment in terms of its benefit (i.e., the accuracy and usefulness of the recovered geological information).

According to [73], a significant portion of the costs involved in seismic surveys is associated with the time and manpower required to deploy and retrieve 'seismic hardware', which includes seismic sensor devices (placed on the surface or in wells), as well as the associated cabling. Therefore, the issue of sensor placement is an important aspect of seismic survey design. Proper sensor placement design should offer an opportunity to reduce the cost of surveys without compromising on the quality of the data acquired. Ideally, during the design of a survey, reliable placement of sensors should be determined using optimal sensor placement techniques. Optimisation of sensor locations is therefore an important and worthwhile problem of interest to the geophysical community, not just in prospecting but also in fault detection and characterisation. Furthermore, as sensor arrays are used in numerous applications, sensor placement is a worthwhile issue in many experimental design problems.

3.1.2 Approaches to Optimisation of Sensor Placement

In designing seismic surveys, the placement of sensors can be chosen by relying on the experience of the operator [113]. However, in the case of complex subsurfaces, more sophisticated strategies have been developed to choose optimal sensor locations. Optimisation procedures for sensor placement have played a significant role in enhancing the quality of data collected during acquisition, while at the same time conserving resources. The details of the methods vary, however, in general, these optimisation algorithms are all driven by the goal of maximising the data information used to characterise the subsurface. This section provides a review of some common mathematical approaches to sensor placement, which include methods using the *Fisher Information* matrix as a tool for sensor optimisation, sequential algorithms to maximise *observability*, *global optimisation* methods, and *Bayesian experimental design*. Since optimal sensor placement is a broad area with many applications, this overview of optimal sensor placement techniques is general and not specific to seismic imaging.

We begin by noting that the mathematical and statistical foundations of optimal experimental design was pioneered by R.A. Fisher in the early 1900's, and this theory has been routinely applied in physical, biological, and social experimental design. The *Fisher Information Matrix* (FIM) quantifies the amount of information that an observation carries about an unknown parameter [28]. Various metrics based on the FIM may be maximised or minimised to achieve the objectives of a survey, for example the condition number, trace and determinant of the FIM. A popular approach to optimal sensor placement involves the maximisation of the determinant of the FIM. This technique has been widely studied and implemented for optimising sensor locations, for example

in [152] and [131]. This maximisation method is adapted by Kammer [101] in the context of Structural Health Monitoring (SHM), with their iterative approach being termed Effective Independence (EI). The EI method requires the prior selection of a set of target features to be identified, and a large set of possible sensor locations. The method ranks the candidate sensor locations in this larger set, according to their spatial independence. At each iteration of the algorithm, the lowest ranking sensor is removed, and this process is continued until the required number of sensors are left. Other methods of ranking the importance of candidate sensor locations exist, including the Modal Kinetic Energy (MKE) method which is common in the area of SHM [115].

This idea of sequentially maximising some measure of the *observability* has been used in seismic imaging. Curtis et al. [53] defines a measure of receiver quality, which depends on the importance of the data that a specified receiver is expected to record. This method assumes that the number of possible sensor placements is finite, and that the model-data relationship is approximately linear, i.e., $\mathbf{d} = G\boldsymbol{m}$ for a sensitivity matrix G. The receiver quality factor is related to the linear independence of the matrix G. At each iteration of the method, the number of sensors is reduced, removing the most redundant sensor at each step. Therefore, the method will only keep sensors that provide information that is as independent as possible from all other sensors. Stummer et al. [172] and Coles et al. [45] use sequential algorithms that work in the opposite direction, beginning from a minimal design and iteratively adding sensors that provide the most informative data at each step. A drawback of these type of methods is that non-linearity is difficult to incorporate [92], (an issue since FWI is strongly non-linear) but have the advantage that they are computationally non-intensive compared to exhaustive search techniques.

Another possible approach to the optimal sensor placement problem includes *global optimisation methods*. Global optimisation methods used in sensor placement problems include, for example, simulated annealing (e.g., for eathquake problems [88], and ocean tomography problems [16]) and genetic algorithms (e.g., for industrial system diagnosis [169] and for structural health monitoring [95]). More details on global optimisation methods can be found in [119]. Global search methods for finding optimal placements can be too slow to be practical for larger scale problems [45]. This suggests the possible use of local optimisation methods, which trade global optimality for speed.

Some methods of survey design and optimal sensor placement assume prior knowledge, for example in *Bayesian* experimental design. An application of Bayesian experimental design to seismic travel-time tomography is found in [103]. The concept of exploiting available information during survey design is important in seismic applications. New seismic data acquisition is still required in areas with previous seismic coverage, due to the need for higher accuracy in oil exploration. Therefore, a lot of prior knowledge about the subsurface may already be available. In order to make new seismic data as valuable as possible and to achieve the best possible return from the seismic exploration process, the prior information about the subsurface should be taken into account in designing the survey and choosing sensor locations. In an area where no exploration has taken place before, this will not be possible. However, in more mature areas, all available subsurface information can be exploited to improve the seismic survey design. It is also possible to use a non-Bayesian approach to designing acquisition

based on prior knowledge of the subsurface, for example the method proposed in [117].

3.2 Thesis Idea for Sensor Placement Optimisation

In this thesis, we investigate the incorporation of prior subsurface information to inform the sensor placement optimisation. The prior information is in the form of training sets of ground truth images/models with corresponding data. We define a sensor placement optimisation objective function that measures the error/misfit between the training subsurface images and the reconstructions of the subsurface produced by FWI, for a specific layout of sensors. We propose a gradient-based optimisation method to iteratively reduce this error, i.e., at each iteration of the optimisation, we require the gradient of the objective function with respect to sensor position, and use this gradient information to move the sensors in such a way that the objective function is reduced. We then compute the reconstruction produced by FWI at this new set of sensor positions. This continues until the error has decreased sufficiently, with the final goal being the enhancement of the FWI reconstruction/imaging results due to the optimal placement of sensors. Since the optimisation method aims to find the sensor positions that will result in the FWI output models reproducing given models as well as possible, this is a *learning* problem, where the best sensor positions are learned from the training models. Later in this chapter we also incorporate the optimisation of a regularisation parameter into this learning problem. We note that since we use a local optimisation method, there is no guarantee that we will find the global minimum, but we can at least find a local minimum and hence an improved set of sensor positions and improved regularisation parameter.

The idea behind our seismic sensor optimisation algorithm is novel since it is combines the approach of *learning* optimal sensor positions with the standard approach to *FWI*. Incorporating FWI into the sensor placement optimisation approach means that we will be dealing with the full non-linear ill-posed seismic imaging problem, which is solved with a local-optimisation method. The FWI reconstruction is used in the measure of error at each step of the local sensor optimisation algorithm. We have seen in the previous section (§3.1.2) that approaches to optimal design often involve defining some measure of how good or bad a design setup is, but none of these approaches use the FWI reconstruction itself in this measure. In addition, many of the other approaches discussed in the previous subsection involve assumptions, such as the linearity of the forward problem, and restrictions, such as a finite set of possible sensor positions. Our method does not require these assumptions or restrictions, as FWI is non-linear and we are searching for optimal sensor locations from an infinite number of possible sensor positions.

3.3 Problem Formulation

Our sensor placement optimisation problem assumes a fixed number of sensors, N_r, and aims to find the optimal locations for these sensors. We formulate the sensor

optimisation problem under the assumption that we have access to a prior model $\boldsymbol{m}'$, or several prior models, each with corresponding data $\mathbf{d}(\boldsymbol{m}')$. This set of prior models will be treated as a set of clean training images/training models, meaning we take them to be ground truth. We define the quantity to be minimised during optimisation as the difference between each training model $\boldsymbol{m}'$ and the FWI reconstruction of that model, $\boldsymbol{m}^{FWI}$, defined in the squared two norm, summed over all training models and scaled by the number of training models. The objective function is therefore a measure of the *average error in the FWI solution* across all training models.

Notation 3.3.1. ***Labelling of Sensors:*** *Before we state the sensor placement objective function, we introduce some new notation. In Chapter 2 we usually referred to sensors in an abstract way with the symbol $\mathcal{P}$, which denotes the set of sensors. In this chapter, we require the optimisation of sensor coordinates/positions, and therefore we need to explicitly identify these coordinates. We define a vector $\boldsymbol{p} = [p_1\ p_2\ \dots\ p_{dN_r}]^T$ which is a concatenated position vector containing all the coordinates of the sensors, where N_r is the number of sensors, d is the spatial dimension of the problem, so that the product dN_r is the number of sensor coordinates. For example, if we are working with a problem in two dimensions, then $dN_r = 2N_r$. Clearly the set of sensor positions $\mathcal{P}$ depends directly on the sensor coordinates $\boldsymbol{p}$ (i.e., these are just different ways of expressing the same information). For clarity in this chapter, we will be writing $\boldsymbol{p}$ instead of $\mathcal{P}$. Therefore, the notation used is slightly different to that in Chapter 2 but the meaning is the same.*

Definition 3.3.2. ***Sensor Placement Objective Function:*** *Let $\mathcal{M}'$ be a set of $N_{m'}$ training models, and $\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')$ be the solution to the FWI problem with sensor positions $\boldsymbol{p}$ for each training model $\boldsymbol{m}' \in \mathcal{M}'$. The sensor placement objective function is*

$$\psi(\boldsymbol{p}) := \frac{1}{2N_{m'}} \sum_{\boldsymbol{m}' \in \mathcal{M}'} ||\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')||_2^2, \tag{3.3.1}$$

where for each training model $\boldsymbol{m}' \in \mathcal{M}'$, the corresponding FWI solution $\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')$ is defined by

$$\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}') = \underset{\boldsymbol{m}}{argmin}\ \phi(\boldsymbol{m}, \boldsymbol{p}, \boldsymbol{m}'). \tag{3.3.2}$$

We define the FWI objective function ϕ in (3.3.2) as,

$$\phi(\boldsymbol{m}, \boldsymbol{p}, \boldsymbol{m}') = \frac{1}{2} \sum_{s \in \mathcal{S}} \sum_{\omega \in \mathcal{W}} \|\boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, \omega, s)\|_2^2 + \frac{1}{2}\mu \|\boldsymbol{m}\|_2^2 \tag{3.3.3}$$

for each $\boldsymbol{m}' \in \mathcal{M}'$, where $\boldsymbol{\varepsilon}$ is defined in (3.3.5) below. In practice, ϕ can include any other regularisation terms, but we do not consider this scenario until later in this chapter (Section 3.5).

We note some points about the notation used in Definition 3.3.2 for clarity. In (3.3.1), for simplicity, we do not explicitly indicate the dependence of the objective function ψ on the set of training models $\mathcal{M}'$, although the sum in (3.3.1) is over all training models $\boldsymbol{m}' \in \mathcal{M}'$. In addition, the argument of ψ is written simply as the sensor positions $\boldsymbol{p}$, but in reality it depends on $\boldsymbol{p}$ through the FWI solutions (3.3.2). The FWI objective function ϕ technically depends on the training model through the data $\mathbf{d}$ (defined below in (3.3.4)), but we write the dependence on $\boldsymbol{m}'$ directly for clarity. Thus we are using slightly different notation in Chapter 3 than in Chapter 2 in terms of the dependencies of ϕ and $\boldsymbol{m}^{FWI}$.

We now make an important point about the observed FWI data $\mathbf{d}$. In the standalone FWI problem (i.e., that described in Chapter 2), we are given observed data that comes from real subsurface wavefield measurements. In the sensor placement optimisation problem, there is a set of data corresponding to each training model $\boldsymbol{m}'$, and this data is fully sampled (sampled at every possible sensor position). We simulate this data from the training models ourselves, i.e., all 'observed' data is actually synthetic in the sensor optimisation problem. Written in continuous variables, the synthetic data is given by the following expression, for each training model $\boldsymbol{m}'$, sensor positions $\boldsymbol{p}$, source $s \in \mathcal{S}$ and frequency $\omega \in \mathcal{W}$,

$$\mathbf{d}(\boldsymbol{m}', \boldsymbol{p}, s, \omega) = \mathcal{R}(\boldsymbol{p}) u(\boldsymbol{m}', s, \omega) + \eta, \tag{3.3.4}$$

where η is a noise term that is added to simulate real FWI where data collected at sensors can be noisy. When the data is defined by (3.3.4), the residual will be a function of the training model, so we write,

$$\boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, \omega, s) = \mathbf{d}(\boldsymbol{m}', \boldsymbol{p}, \omega, s) - \mathcal{R}(\boldsymbol{p}) u(\boldsymbol{m}, \omega, s). \tag{3.3.5}$$

In Remark 3.3.3, we describe how to compute the data in practice.

Remark 3.3.3. ***Observed Data for the Discrete Problem:*** *Here we write the synthetic data in discrete variables, since this describes how it is computed in practice,*

$$\boldsymbol{d}(\boldsymbol{m}', \boldsymbol{p}, s, \omega) = R(\boldsymbol{p}) \boldsymbol{u}(\boldsymbol{m}', s, \omega) + \eta, \tag{3.3.6}$$

i.e., the forward problem (2.2.14) is solved for the training model $\boldsymbol{m}'$, the restriction matrix is applied to extract the values of the wavefield $\boldsymbol{u}(\boldsymbol{m}')$ at the sensor positions, and noise can be added to the data to simulate real noisy FWI data. The residual written in discrete variables is

$$\boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, \omega, s) = \boldsymbol{d}(\boldsymbol{m}', \boldsymbol{p}, \omega, s) - R(\boldsymbol{p}) \boldsymbol{u}(\boldsymbol{m}, \omega, s). \tag{3.3.7}$$

In implementation, as we wish to avoid an inverse crime, the wavefields for the 'observed' data (i.e., $\boldsymbol{u}(\boldsymbol{m}')$) and the modelled data (i.e., $\boldsymbol{u}(\boldsymbol{m})$) are computed on different grids and so the discrete restriction operators are different for the 'observed' and modelled data (i.e., the $R(\boldsymbol{p})$ in (3.3.6) and the $R(\boldsymbol{p})$ in (3.3.7) are on different grids).

Note that since (3.3.6) is synthetic data that we compute ourselves, we can also choose to consider the simpler case where $\eta = 0$. In this case, the data is just the exact output from a forward solve at the sensor positions. We remark here that all later experiments in this thesis involve setting η to 0.

We now make the full problem statement for our sensor placement optimisation problem.

Definition 3.3.4. ***Sensor Placement Optimisation Problem:***

$$\text{Find} \quad \boldsymbol{p}_{\min} = \underset{\boldsymbol{p}}{\operatorname{argmin}} \ \psi(\boldsymbol{p}) \tag{3.3.8}$$

$$\textit{subject to} \quad \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}') = \underset{\boldsymbol{m}}{\operatorname{argmin}} \ \phi(\boldsymbol{m}, \boldsymbol{p}, \boldsymbol{m}') \quad \textit{for each } \boldsymbol{m}' \in \mathcal{M}' \tag{3.3.9}$$

where ψ is defined by (3.3.1) and ϕ is defined by (3.3.3).

We can see from Definition 3.3.4 and Equation (3.3.1), that ψ depends on the FWI solution $\boldsymbol{m}^{FWI}$, meaning that there is an optimisation problem embedded in another optimisation problem. This embedding gives the problem the structure of a *bilevel optimisation* problem. A bilevel optimisation problem differs from a conventional optimisation problems in that one of its constraints also involves an optimisation problem. The main optimisation task of the bilevel problem is called the *upper-level* optimisation problem and the optimisation problem in the set of constraints is called the *lower-level* optimisation. In our bilevel problem, the upper-level is the sensor placement optimisation problem (learning optimal sensor placements from the training set) and the lower-level is FWI (producing a reconstruction of the subsurface).

Appendix I provides a general review on bilevel optimisation.

3.4 Analysis of Bilevel Optimisation Problem

We choose to solve the sensor optimisation bilevel problem using *single-level reduction* and a *gradient-based optimisation method*. This solution approach is detailed for general bilevel optimisation problems in Appendix I. We make various assumptions that will allow us to solve our bilevel problem in this way. These assumptions are required for the analysis in Sections 3.4.1 and 3.4.2, and justified in Section 3.4.3. We note that these assumptions are required for the theory, but are not checked in practice in our later computations.

Assumption 3.4.1.

(i) The lower-level objective function ϕ is twice continuously differentiable with respect to the model $\boldsymbol{m}$.

(ii) The lower-level gradient $\nabla_{\boldsymbol{m}}\phi$ with respect to the model is continuously differentiable with respect to sensor position $\boldsymbol{p}$.

(iii) The conditions on the regularisation parameter in Theorem 2.4.18 are satisfied; this has the following two consequences:

(a) The lower-level Hessian is positive definite and hence invertible because, as discussed at the beginning of Section 2.4.4, the choice of regularisation prevents a singular Hessian.

(b) By Corollary 2.4.20, a unique solution exists to the lower-level problem. This uniqueness makes the problem formulation in Definition 3.3.4 well-defined.

This section provides details on the single-level reformulation (§3.4.1), a derivation of the gradient of the upper-level with respect to sensor positions (§3.4.2), and analysis and proofs of some of the above assumptions (§3.4.3).

3.4.1 Single-Level Reduction

Single-level reduction is an approach that transforms a bilevel optimisation problem into a single-level optimisation problem. The reduction of the bilevel problem to a single level makes the problem easier for conventional optimisation methods. Details of this approach can be found in [61, Section 4.3] and [163, Section III A].

We transform the bilevel problem (Definition 3.3.4) into a single-level problem by replacing the lower-level problem with its necessary optimality condition. The reduced single-level problem can then be solved with a gradient-based optimisation method. The single-level problem can be written as

$$\begin{aligned} \text{Find} \quad & \boldsymbol{p}_{\min} = \operatorname*{argmin}_{\boldsymbol{p}} \psi(\boldsymbol{p}) \\ \text{subject to} \quad & \nabla_{\boldsymbol{m}}\phi(\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}'), \boldsymbol{p}, \boldsymbol{m}') = \boldsymbol{0} \quad \text{for each } \boldsymbol{m}' \in \mathcal{M}', \end{aligned} \tag{3.4.1}$$

where ψ is given in (3.3.1) and ϕ is given in (3.3.3). In other words, for each training model $\boldsymbol{m}'$, the FWI solution $\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')$ is characterised by by requiring it to be a critical point of ϕ with respect to $\boldsymbol{m}$. This is necessary but not sufficient for (3.3.9).

With the problem in the form (3.4.1), we derive formulae for the upper-level gradient of the sensor optimisation problem in the following section.

3.4.2 Gradient of Sensor Placement (Upper-Level) Objective Function

To apply a gradient-based optimisation method to the sensor placement problem, we require the computation of the gradient of the upper-level objective function with respect to sensor position $\boldsymbol{p}$. In this section we derive a formula for this upper-level gradient, present an algorithm for its computation, and discuss the cost of computation in terms of the number of PDE solves required.

Theorem 3.4.2 presents the formula and derivation of the upper-level gradient. It is worth noting that the parametrisation can be generalised to constrain the learned positions (for example, we could optimise one coordinate of the sensors while keeping the other coordinates fixed) but we don't consider the constrained case in the derivation of Theorem 3.4.2.

Theorem 3.4.2. ***Upper-Level Gradient with respect to p:*** *Under Assumption 3.4.1, the gradient of the upper-level objective function ψ can be written as*

$$\nabla_{\boldsymbol{p}}\psi(\boldsymbol{p}) = \frac{1}{N_{m'}} \sum_{\boldsymbol{m}'\in\mathcal{M}'} \left[\Big(B(\boldsymbol{m}^{FWI}, \boldsymbol{p})\Big)^T H(\boldsymbol{m}^{FWI}, \boldsymbol{p})^{-1} \left(\boldsymbol{m}' - \boldsymbol{m}^{FWI}\right) \right] \tag{3.4.2}$$

where $\boldsymbol{m}^{FWI} = \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')$ and $H \in \mathbb{R}^{M\times M}$ is the Hessian (defined by (2.4.11) with the addition of the term μI coming from regularisation), and $B \in \mathbb{R}^{M\times dN_r}$ is defined by

$$B_{kn}(\boldsymbol{m}, \boldsymbol{p}) = \frac{\partial^2 \phi(\boldsymbol{m}, \boldsymbol{p})}{\partial p_n \partial m_k} \tag{3.4.3}$$

$$= -\sum_{s\in\mathcal{S}} \sum_{\omega\in\mathcal{W}} \Re\left\{ \left(\frac{\partial u(\boldsymbol{m}, s, \omega)}{\partial m_k}, \frac{d}{dp_n}\Big(\mathcal{R}(\boldsymbol{p})^* \boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, s, \omega)\Big) \right) \right\} \tag{3.4.4}$$

for $k = 1, ..., M$ and $n = 1, ..., dN_r$.

Proof. In this proof, we assume only one training model $\boldsymbol{m}'$, i.e., $N_{m'} = 1$, so that we can drop the summation over $\boldsymbol{m}'$ in (3.3.1) and the dependencies of all variables on $\boldsymbol{m}'$ for clarity. We also assume only one source s and one frequency ω and will drop the dependencies on these also.
The first step is to differentiate ψ in (3.3.1) with respect to p_n, for $n = 1, ..., dN_r$, to obtain,

$$\frac{\partial \psi(\boldsymbol{p})}{\partial p_n} = \left\langle -\frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p})}{\partial p_n}, (\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p})) \right\rangle. \tag{3.4.5}$$

The first term in the inner product (3.4.5) is the derivative of the reconstructed image, $\boldsymbol{m}^{FWI}$, with respect to the nth sensor coordinate. (Note that we justify the differentiability of $\boldsymbol{m}^{FWI}$ with respect to sensor position in Section 3.4.3.) To find an expression for the term $\partial \boldsymbol{m}^{FWI}/\partial p_n$, recall the constraint (3.4.1) which, in our simplified notation, reads,

$$\nabla_{\boldsymbol{m}} \phi(\boldsymbol{m}^{FWI}(\boldsymbol{p}), \boldsymbol{p}) = \boldsymbol{0}. \tag{3.4.6}$$

Taking the total derivative of (3.4.6) with respect to p_n, i.e.,

$$\frac{d}{dp_n}\Big(\nabla_{\boldsymbol{m}} \phi(\boldsymbol{m}^{FWI}(\boldsymbol{p}), \boldsymbol{p})\Big) = \boldsymbol{0},$$

we obtain, via the chain rule, the following linear system,

$$H(\boldsymbol{m}^{FWI}(\boldsymbol{p}),\boldsymbol{p})\,\frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p})}{\partial p_n} = -\frac{\partial^2\phi(\boldsymbol{m}^{FWI}(\boldsymbol{p}),\boldsymbol{p})}{\partial p_n \partial \boldsymbol{m}}, \tag{3.4.7}$$

where H is the Hessian of ϕ with respect to $\boldsymbol{m}$, and the right-hand side of (3.4.7) is the nth column of B (defined in (3.4.3)), which we will denote here as $\mathbf{b}_n \in \mathbb{R}^{M\times 1}$. By Assumption 3.4.1 (iii), the solution of (3.4.7) is as follows,

$$\frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p})}{\partial p_n} = -\left(H(\boldsymbol{m}^{FWI}(\boldsymbol{p}),\boldsymbol{p})\right)^{-1}\mathbf{b}_n \tag{3.4.8}$$

(Note that we rigorously justify the formula (3.4.8) in Corollary 3.4.30.) Substituting (3.4.8) into (3.4.5) gives the nth element of the gradient

$$\frac{\partial \psi(\boldsymbol{p})}{\partial p_n} = \left\langle \left(H(\boldsymbol{m}^{FWI}(\boldsymbol{p}),\boldsymbol{p})\right)^{-1}\mathbf{b}_n, (\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p}))\right\rangle$$

Therefore, using the symmetry of the Hessian, we have the following gradient formula, for all sensor positions $n = 1, ..., dN_r$,

$$\begin{aligned}\nabla_{\boldsymbol{p}}\psi(\boldsymbol{p}) &= \left(H(\boldsymbol{m}^{FWI}(\boldsymbol{p}),\boldsymbol{p})^{-1}B(\boldsymbol{m}^{FWI}(\boldsymbol{p}),\boldsymbol{p})\right)^T(\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p})),\\ &= B(\boldsymbol{m}^{FWI}(\boldsymbol{p}),\boldsymbol{p})^T H(\boldsymbol{m}^{FWI}(\boldsymbol{p}),\boldsymbol{p})^{-1}(\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p})).\end{aligned} \tag{3.4.9}$$

The extension of the result (3.4.9) for many training models is (3.4.2).

To obtain the formula (3.4.4), we differentiate the kth entry of the gradient (2.3.1) with respect to p_n. Using identity (2.2.19),

$$\begin{aligned}\frac{\partial}{\partial p_n}\left(\frac{\partial\phi(\boldsymbol{m},\boldsymbol{p})}{\partial m_k}\right) &= -\Re\left\{\frac{\partial}{\partial p_n}\left[\left\langle \mathcal{R}(\boldsymbol{p})\frac{\partial u(\boldsymbol{m})}{\partial m_k}, \boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{p})\right\rangle\right]\right\} + \frac{\partial(\mu m_k)}{\partial p_n}\\ &= -\Re\left\{\frac{\partial}{\partial p_n}\left[\left(\frac{\partial u(\boldsymbol{m})}{\partial m_k}, \mathcal{R}(\boldsymbol{p})^*\boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{p})\right)\right]\right\}\\ &= -\Re\left\{\left(\frac{\partial u(\boldsymbol{m})}{\partial m_k}, \frac{d}{dp_n}\left(\mathcal{R}(\boldsymbol{p})^*\boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{p})\right)\right)\right\}.\end{aligned} \tag{3.4.10}$$

Note that the derivative of the convex regularisation term vanishes here because it has no dependence on sensor position. When summed over sources and frequencies, (3.4.10) is equal to (3.4.4). ∎

Remark 3.4.3. ***Linear System appearing in the Gradient (3.4.2):*** *Computing* (3.4.2) *involves solving the following linear system, for each training model* $\boldsymbol{m}'$*,*

$$H(\boldsymbol{m}^{FWI}(\boldsymbol{p},\boldsymbol{m}'),\boldsymbol{p})\boldsymbol{\delta}(\boldsymbol{m}^{FWI}(\boldsymbol{p},\boldsymbol{m}'),\boldsymbol{m}') = \boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p},\boldsymbol{m}'), \tag{3.4.11}$$

and then inserting the solution $\boldsymbol{\delta}$ *into* (3.4.2) *to give*

$$\nabla_{\boldsymbol{p}}\psi(\boldsymbol{p}) = \frac{1}{N_{m'}}\sum_{\boldsymbol{m}'\in\mathcal{M}'}\left[\left(B(\boldsymbol{m}^{FWI}(\boldsymbol{p},\boldsymbol{m}'),\boldsymbol{p})\right)^T\boldsymbol{\delta}(\boldsymbol{m}^{FWI}(\boldsymbol{p},\boldsymbol{m}'),\boldsymbol{m}')\right]. \tag{3.4.12}$$

In practice, the gradient is computed with Equation (3.4.12) by: (i) solving (3.4.11) for $\boldsymbol{\delta}$ for each $\boldsymbol{m}'$, (ii) computing the matrix B for each $\boldsymbol{m}'$, (iii) performing the multiplication in (3.4.12) explicitly and summing over the training models.

Remark 3.4.4. ***Cost of Solving the Linear System (3.4.11):*** *As discussed in Section 2.4, computing the Hessian explicitly requires M+2 PDE solves, however computing a general Hessian-vector product requires only 4 PDE solves, 2 of which involve the vector and 2 which are independent of the vector (see Theorem 2.4.3). Therefore, (3.4.11) should be solved with a method that requires only Hessian-vector products, such as the conjugate gradient method (assuming the Hessian is positive definite). Solving (3.4.11) using such a matrix-free algorithm involves solving (2+2N_i) PDEs, where N_i is the number of iterations of the chosen solution method (see [129, Section 3.3] for more detail). However, for the specific system (3.4.11), 2 of these PDEs will already have been solved to compute $\boldsymbol{m}^{FWI}$, and so the computation of $\boldsymbol{\delta}$ involves only $2N_i$ PDE solves for each source, frequency and training model.*

Remark 3.4.5. ***An Interpretation of the Linear System (3.4.11):*** *Equation (3.4.11) gives a formula for $\boldsymbol{\delta}$ which involves the inverse of the Hessian evaluated at the FWI solution. In the Bayesian approach to the FWI problem, it can be shown that, under the Laplace approximation (see [26, Section 4.4]), the inverse of the Hessian evaluated at the FWI solution can be interpreted as the posterior covariance matrix for the FWI/lower-level problem. The other term in the formula for $\boldsymbol{\delta}$ is the discrepancy, or misfit, between the training model $\boldsymbol{m}'$ and the FWI solution $\boldsymbol{m}^{FWI}$. This misfit can be interpreted as the error in the FWI solution.*

We now discuss the computation of the matrix B defined in (3.4.4). An initial inspection of (3.4.4) suggests that computing B will involve M PDE solves, to compute each partial derivative wavefield $\partial u/\partial m_k$ (given by (2.3.3)). However, similar to Theorem 2.3.1, we do not require the explicit computation of $\partial u/\partial m_k$ in (3.4.4), only inner-products involving $\partial u/\partial m_k$. Applying the adjoint-state method reduces the cost of computing B to a total of $dN_r + 1$ PDE solves for each source and each frequency - of which dN_r PDE solves are to compute the adjoint variable γ_n, defined in (3.4.14), and one additional PDE solve is needed to compute the forward wavefield u. This result is written as Theorem 3.4.6 below. Note that the matrix B in the gradient expression (3.4.2) is evaluated at the FWI solution $\boldsymbol{m}^{FWI}$ so the forward wavefield will already have been computed and stored.

Theorem 3.4.6. ***Adjoint-State Method for evaluating B (3.4.4):*** *The matrix* $B \in \mathbb{R}^{M \times dN_r}$ *(3.4.4), appearing in the expression for the upper-level gradient (3.4.2) may be written as*

$$B_{kn}(\boldsymbol{m}, \boldsymbol{p}) = -\Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}} \omega^2 \left(u(\boldsymbol{m}, s, \omega), \gamma_n(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, s, \omega)\right)_{\beta_k}\right\} \tag{3.4.13}$$

for $k = 1, ...M$*, where* γ_n *is defined as the solution to*

$$\gamma_n(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, s, \omega) = \mathscr{S}^*_{\boldsymbol{m},\omega}\left(\frac{d}{dp_n}\Big(\mathcal{R}(\boldsymbol{p})^*\boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, s, \omega)\Big)\right) \tag{3.4.14}$$

for $n = 1, .., dN_r$*.*

Proof. For simplicity in this proof, we assume one source and one frequency, and so the dependencies that are written in (3.4.13) and (3.4.14) will be dropped for simplicity.

Substituting the expression (2.3.3) for $\partial u/\partial m_k$ into (3.4.4), we find that

$$\begin{aligned} B_{kn} &= -\Re\left\{\left(\omega^2\mathscr{S}_{\boldsymbol{m}}(\beta_k u(\boldsymbol{m})), \frac{d}{dp_n}\Big(\mathcal{R}(\boldsymbol{p})^*\boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p})\Big)\right)\right\} \\ &= -\omega^2\Re\left\{\left(\beta_k u(\boldsymbol{m}), \mathscr{S}^*_{\boldsymbol{m}}\left(\frac{d}{dp_n}\Big(\mathcal{R}(\boldsymbol{p})^*\boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p})\Big)\right)\right)\right\} \end{aligned}$$

Defining γ_n as in (3.4.14), the (k, n)th element of B can be written as

$$B_{kn} = -\omega^2\Re\left\{\left(u(\boldsymbol{m}), \gamma_n(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p})\right)_{\beta_k}\right\}.$$

In the case of multiple sources and frequencies involves summations as written in (3.4.13). ∎

Remark 3.4.7. ***Comparison of Lower- and Upper-Level Gradients:*** *We can directly compare the expressions for the FWI (lower-level) gradient (2.3.4) and its adjoint variable (2.3.5) with the expressions just derived for B (3.4.13) and the upper-level adjoint variable (3.4.14). Both (2.3.4) and (3.4.13) have the same form in that they are both weighted inner-products of the forward wavefield u and an adjoint wavefield. We can also directly compare the adjoint wavefield* λ*, which is the solution to the wave equation (2.3.5), and the adjoint wavefields* γ_n *for* $n = 1, \ldots, dN_r$*, which are the solution to the wave equations (3.4.14). We can see that the only difference between these wave equations are the right-hand sides. Indeed, the right-hand side of (3.4.14) is simply the derivative of the right-hand side of (2.3.5) with respect to the sensor coordinates.*

Remark 3.4.8. ***Right-Hand Side of (3.4.14):*** *The right-hand side of* (3.4.14) *is interesting because it involves the derivative of the delta function. Here we discuss the*

meaning of the right-hand side of (3.4.14) *in the one-dimensional case for simplicity. In the one-dimensional case* p_n *means the only coordinate of the* n*th sensor, i.e.,* $N_r = dN_r$*. The same discussion also applies for higher dimensions. Recall the definition* (2.2.18) *of the adjoint of the restriction operator,* $\mathcal{R}(\boldsymbol{p})^*$*. Using* (2.2.18)*, the right-hand side of* (3.4.14) *can be written as*

$$\frac{d}{dp_n}\Big(\mathcal{R}(\boldsymbol{p})^*\boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{m}',\boldsymbol{p})\Big) = \frac{d}{dp_n}\left(\sum_{j=1}^{N_r}\delta_{p_j}\boldsymbol{\varepsilon}_j(\boldsymbol{m},\boldsymbol{m}',\boldsymbol{p})\right). \tag{3.4.15}$$

where we have suppressed the dependence on s *and* ω *for simplicity. We note that, by definition of the residual* (3.3.5) *and data* (3.3.4)*, in one-dimension the* j*th entry of the residual is given by*

$$\boldsymbol{\varepsilon}_j(\boldsymbol{m},\boldsymbol{m}',\boldsymbol{p}) = u(\boldsymbol{m}';p_j) + \eta - u(\boldsymbol{m};p_j), \tag{3.4.16}$$

where we are writing $(\mathcal{R}(\boldsymbol{p})u(\boldsymbol{m}))_j$ *as* $u(\boldsymbol{m};p_j)$*. We see that the* j*th entry of the residual depends on the sensor with coordinate* p_j *only, and hence the derivative* (3.4.15) *vanishes unless* $n = j$*. Using this fact and applying the product rule to* (3.4.15) *gives,*

$$\frac{d}{dp_n}\Big(\mathcal{R}(\boldsymbol{p})^*\boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{m}',\boldsymbol{p})\Big) = \frac{d\delta_{p_n}}{dp_n}\boldsymbol{\varepsilon}_n(\boldsymbol{m},\boldsymbol{m}',\boldsymbol{p}) + \delta_{p_n}\frac{\partial\boldsymbol{\varepsilon}_n(\boldsymbol{m},\boldsymbol{m}',\boldsymbol{p})}{\partial p_n}. \tag{3.4.17}$$

If the noise term in (3.4.16) *is assumed not to depend on sensor position, then the derivative of the* n*th component of the residual term is*

$$\frac{\partial\boldsymbol{\varepsilon}_n(\boldsymbol{m},\boldsymbol{m}',\boldsymbol{p})}{\partial p_n} = \frac{\partial u(\boldsymbol{m}';p_n)}{\partial p_n} - \frac{\partial u(\boldsymbol{m};p_n)}{\partial p_n}.$$

We note that by definition of the derivative of the delta function and the fact that $\delta_{p_n} = \delta(x - p_n)$*, for any function* v*,*

$$\frac{d\delta_{p_n}}{dp_n}v = -(-\delta_{p_n})\frac{dv}{dp_n} = \frac{dv(p_n)}{dp_n}.$$

This remark explains the meaning of the right-hand side of (3.4.14) *in the continuous case; in Remark 3.4.12 we give details of its discrete analogue.*

Algorithm 3.4.9. ***Algorithm for Computing the Upper-Level Gradient:*** *The computation of the upper-level gradient $\nabla_{\boldsymbol{p}}\psi$ involves the following steps. For a given set of set of sensor positions with sensor coordinates $\boldsymbol{p}$, a set of training models $\mathcal{M}'$, and corresponding set of FWI solutions $\boldsymbol{m}^{FWI} = \boldsymbol{m}^{FWI}(\boldsymbol{p}, \mathcal{M}')$:*

- *For each $\boldsymbol{m}' \in \mathcal{M}'$:*
  - *For each $\omega \in \mathcal{W}$, $s \in \mathcal{S}$:*
    - *Compute the FWI data $\boldsymbol{d}(\boldsymbol{m}', \boldsymbol{p}, s, \omega)$ by (3.3.4)*
    - *Compute $u(\boldsymbol{m}^{FWI}, \omega, s)$ via the forward problem (2.2.5)*
    - *Compute residual $\boldsymbol{\varepsilon}(\boldsymbol{m}^{FWI}, \boldsymbol{m}', \boldsymbol{p}, \omega, s)$ via (3.3.5)*
    - *For each optimisation variable $n = 1, .., dN_r$:*
      - *Compute each $\gamma_n(\boldsymbol{m}^{FWI}, \boldsymbol{m}', \boldsymbol{p}, s, \omega)$ via (3.4.14)*
      - *Compute the nth column of $B(\boldsymbol{m}^{FWI}, \boldsymbol{p})$ for each source and frequency via (3.4.13) for all $k = 1, ..., M$*
  - *Assemble B by summing over $s \in \mathcal{S}$ and $\omega \in \mathcal{W}$*
  - *Solve (3.4.11) for $\boldsymbol{\delta}(\boldsymbol{m}^{FWI}, \boldsymbol{m}')$*
- *Compute the gradient $\nabla_{\boldsymbol{p}}\psi \in \mathbb{R}^{dN_r}$ by (3.4.12)*

Theorem 3.4.10. ***Cost of Bilevel Gradient Descent:*** *The total cost of solving the bilevel sensor optimisation problem with gradient descent in terms of the number of PDE solves is*

$$\textbf{\textit{Number of PDE Solves}} = \boldsymbol{N_{m'} N_u N_s N_\omega \left(2N_l + 2N_i + dN_r\right)}, \tag{3.4.18}$$

where we use the following notation

- *N_u = number of upper-level iterations*
- *N_l = number of lower-level iterations (FWI will take a different number of iterations on each upper-level iteration, but for an estimate, we assume the number of FWI iterations is always the same).*
- *N_i = number of iterations of the matrix-free algorithm used to solve (3.4.11) (again we make an approximation that this number of iterations remains constant throughout).*

As usual, N_s is the number of sources, N_ω is the number of frequencies, $N_{m'}$ is the number of training models and dN_r is the number of optimisation variables (sensor coordinates). Note that when using line search with gradient descent, there is an additional factor of the number of line search iterations.

Proof. The cost of solving the bilevel problem with gradient descent may be broken down into the following parts:

- Computing $\boldsymbol{m}^{FWI}$ involves performing FWI, which takes $\boldsymbol{2N_lN_sN_\omega}$ PDE solves (since each iteration of FWI using a gradient-based optimisation method requires a forward and adjoint solve for each source and each frequency, as discussed in Section 2.3).
- We have previously stated in Remark 3.4.4, that we require $2 + 2N_i$ PDE solves to solve (3.4.11), for each source and each frequency, i.e., $(2 + 2N_i)N_sN_\omega$ in total. Note that $2N_sN_\omega$ of these PDE solves are already performed on the last iteration of FWI. Therefore, solving the system (3.4.11) takes $\boldsymbol{2N_iN_sN_\omega}$ PDE solves.
- It takes dN_r PDE solves to compute the adjoint variables in (3.4.14). Due to the sum over sources and frequencies in (3.4.13), there is a total of $\boldsymbol{dN_rN_sN_\omega}$ required to compute B. Note that the forward variable u evaluated at $\boldsymbol{m}^{FWI}$ in (3.4.13) will already have been computed for each each source and each frequency on the last iteration of FWI.

Each of the computations in these points need to be repeated for all $N_{m'}$ training models, and on each of the N_u upper iterations. Therefore the total cost of solving the bilevel sensor optimisation problem with gradient descent is given by (3.4.18). ∎

We note that the complexity analysis in Theorem 3.4.10 has been refined and a new result will be given in the paper [64].

Remark 3.4.11. ***Discretisation of the Upper-Level Gradient:*** *With the forward problem (2.2.14), the discretised version of the upper-level gradient (3.4.12) is*

$$\nabla_{\boldsymbol{p}}\psi(\boldsymbol{p}) = \frac{1}{N_{m'}} \sum_{\boldsymbol{m}' \in \mathcal{M}'} B(\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}'), \boldsymbol{p})^T \boldsymbol{\delta}(\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}'), \boldsymbol{m}'). \tag{3.4.19}$$

where $\boldsymbol{\delta}$ is the solution to

$$H(\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}'), \boldsymbol{p})\boldsymbol{\delta}(\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}'), \boldsymbol{m}') = \boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}'), \tag{3.4.20}$$

where discretised Hessian H is given by (2.4.16) plus the μI term coming from the regularisation. The discretised version of (3.4.13) and (3.4.14) are

$$B_{kn}(\boldsymbol{m}, \boldsymbol{p}) = \Re\left\{ \sum_{s \in \mathcal{S}} \sum_{\omega \in \mathcal{W}} \left[\left(\frac{\partial A(\boldsymbol{m}, \omega)}{\partial m_k} \boldsymbol{u}(\boldsymbol{m}, s, \omega) \right)^* \boldsymbol{\gamma}_n(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, s, \omega) \right] \right\} \tag{3.4.21}$$

for $k = 1, ...M$, where $\boldsymbol{\gamma}_n \in \mathbb{R}^{M \times 1}$ is defined as the solution to

$$A(\boldsymbol{m}, \omega)^* \boldsymbol{\gamma}_n(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, s, \omega) = \frac{d}{dp_n}\Big(R(\boldsymbol{p})^* \boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, s, \omega) \Big). \tag{3.4.22}$$

for $n = 1, .., dN_r$.

Remark 3.4.12. ***Discretisation of the Right-Hand Side of the Adjoint Equation (3.4.14):*** *We discuss the discretised version of the right-hand side of the adjoint equation (3.4.14), i.e., the right-hand side of (3.4.22). This remark is the discrete version of Remark 3.4.7. By the product rule, the right-hand side of (3.4.22) is*

$$\frac{d}{dp_n}\Big(R(\boldsymbol{p})^*\boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{m}',\boldsymbol{p})\Big) = \left(\frac{dR(\boldsymbol{p})}{dp_n}\right)^*\boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{m}',\boldsymbol{p}) + R(\boldsymbol{p})^*\frac{\partial\boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{m}',\boldsymbol{p})}{\partial p_n}, \quad (3.4.23)$$

where we have suppressed the dependence on s and ω as they are not relevant to this discussion. We see that (3.4.23) corresponds to the continuous version (3.4.17). As described in Remark 2.2.6, the discretised restriction operator, is an $N_r \times N$ interpolant matrix, where the jth row depends on the jth sensor. Therefore, $dR(\boldsymbol{p})/dp_n$ is just a $N_r \times N$ matrix with non-zero entries in only one row (the row corresponding to the sensor with coordinate p_n). By the definition of the residual (3.3.7), the derivative of the residual with respect to p_n is given by

$$\frac{\partial\boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{m}'\boldsymbol{p})}{\partial p_n} = \frac{\partial\boldsymbol{d}(\boldsymbol{m}',\boldsymbol{p})}{\partial p_n} - \frac{dR(\boldsymbol{p})}{dp_n}\boldsymbol{u}(\boldsymbol{m}). \quad (3.4.24)$$

If we assume the noise term in (3.3.6) is independent of the sensor position, then the derivative of the synthetic data with respect to sensor position is simply

$$\frac{\partial\boldsymbol{d}(\boldsymbol{m}',\boldsymbol{p})}{\partial p_n} = \frac{dR(\boldsymbol{p})}{dp_n}\boldsymbol{u}(\boldsymbol{m}'). \quad (3.4.25)$$

As described in Remark 3.3.3, since we avoid inverse crime in practice, $\boldsymbol{u}(\boldsymbol{m})$ and $\boldsymbol{u}(\boldsymbol{m}')$ are computed on different grids, and so the restriction operator in the data term (3.4.25) is a different size than the other restriction operators in (3.4.23) and (3.4.24).

3.4.3 Smoothness of $\boldsymbol{m}^{FWI}$ with respect to Sensor Position

In this section, we show that the upper-level objective function ψ is smooth with respect to sensor positions $\boldsymbol{p}$. This is an important result as it shows that a gradient-based local optimisation method can be applied successfully to find a solution to the sensor placement problem. By (3.3.1), we see that in order to show that ψ is smooth with respect to $\boldsymbol{p}$, we can just show that $\boldsymbol{m}^{FWI}$ is smooth with respect to $\boldsymbol{p}$. Therefore, showing that $\boldsymbol{m}^{FWI}$ is smooth with respect to $\boldsymbol{p}$ is the central focus of this section.

The main result in this section is that $\boldsymbol{m}^{FWI}$ is a C^1 function of $\boldsymbol{p}$. This is proved in Corollary 3.4.30 and uses the Implicit Function Theorem (IFT) applied to the system

$$\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m},\boldsymbol{p}) = \mathbf{0}, \quad (3.4.26)$$

provided certain assumptions are met. We suppress the dependence of ϕ (and all other variables) on $\boldsymbol{m}'$ in the notation for this section since we consider only one training model here. The general form of the IFT, along with its assumptions, are written in Appendix J. Relating Theorem J-1 to our sensor optimisation problem, we see that in our case $\mathbf{f} = \nabla_{\boldsymbol{m}}\phi$, $\mathbf{x} = \boldsymbol{p}$, $\mathbf{y} = \boldsymbol{m}$, $J = H$ (where H is the Hessian of ϕ) and the function $\mathbf{g}(\mathbf{x}) = \boldsymbol{m}^{FWI}(\boldsymbol{p})$ is the solution to (3.4.26) for a given $\boldsymbol{p}$. Therefore the assumptions we require to apply the IFT to our problem are:

- The lower-level gradient $\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m}, \boldsymbol{p})$ must be continuously differentiable with respect to sensor coordinates $\boldsymbol{p}$ and model $\boldsymbol{m}$ (which is given by Assumptions 3.4.1 (i) and (ii)).
- The Hessian with respect to $\boldsymbol{m}$ evaluated at the FWI solution, i.e., $H(\boldsymbol{m}^{FWI}(\boldsymbol{p}), \boldsymbol{p})$, must be invertible (which is Assumption 3.4.1 (iii)).

By Theorem 2.4.18, we can choose the regularisation parameter μ in (3.3.3) large enough to ensure the Hessian is positive definite for all models and sensor positions. The second assumption is therefore satisfied. So it only remains to show the first assumption, namely that $\nabla_{\boldsymbol{m}}\phi$ is a smooth enough function of $\boldsymbol{p}$ and $\boldsymbol{m}$.

Before we begin these proofs, we state some useful information on the analytic setting. Most of this information can be found in [33], and when the material isn't there, we give specific references.

Definition 3.4.13. ***Sobolev Spaces****:*

$$H^n(\Omega) = \left\{u : D^j u \in L^2(\Omega) \text{ for } j = 0, \ldots, n\right\}.$$

The definition of $H^n(\Omega)$ can also be extended to allow $n \in \mathbb{R}^+$ (i.e., n not necessarily an integer).

Proposition 3.4.14. *If $u \in L^\infty$, and $v \in L^2$ then $uv \in L^2$.*

In this section, we consider two choices of finite element basis functions - piecewise constant and continuous piecewise linear functions. We give details of these in Proposition 3.4.15.

Proposition 3.4.15. ***Sobolev Spaces of Finite Element Functions:***

- *The space of piecewise constant finite element functions is a subspace of $H^r(\Omega)$ for $r < \frac{1}{2}$ [56, Theorem 4.2].*
- *The space for continuous piecewise linear functions is a subspace $H^r(\Omega)$ for $r < \frac{3}{2}$ [56, Theorem 4.1].*

Propositions 3.4.16 and 3.4.17 provide some details on Sobolev functions that we use in the proofs of later results.

Proposition 3.4.16. ***Sobolev Embedding Theorem (Relationship of Sobolev Spaces to Continuous Functions):*** *If $u \in H^{\frac{d}{2}+\varepsilon}$ for any $\varepsilon > 0$, where d is the spatial dimension, then u is continuous [125, Theorem 3.26].*

Proposition 3.4.17. ***Multiplication of Sobolev Functions:*** *If $n_1 \geq n_2$ and $n_1 > \frac{d}{2}$, with $v_1 \in H^{n_1}(\Omega)$ and $v_2 \in H^{n_2}(\Omega)$, then $v_1 v_2 \in H^{n_2}(\Omega)$ [21].*

We present the following results about the Helmholtz equation (with an L^2 source) and Laplace's equation, as these results are combined later to prove properties about the solution to the Helmholtz equation with a point source. In particular, we are working towards showing that the solution of the Helmholtz equation with a point source is square integrable everywhere in Ω, but is much smoother in a subdomain of Ω that does not include the point source.

Proposition 3.4.18. ***Well-posedness of the Helmholtz Equation with Impedance Boundary Conditions:*** *For a bounded Lipschitz domain Ω, if $-(\Delta + \omega^2 m)u = f \in L^2(\Omega)$ with $(\partial_n - i\omega)u = g \in L^2(\partial\Omega)$ and $m \in L^\infty$, then the Helmholtz problem has a unique solution $u \in H^1(\Omega)$ [79, Corollary 2.5].*

Proposition 3.4.19. ***Smoothness of Solutions of Laplace's Equation:*** *Suppose Ω is either a convex polygon or smooth.*

(i) If $\Delta v = f \in L^2(\Omega)$ with $(\partial_n - i\omega)v = g \in C^1(\partial\Omega)$, then $v \in H^2(\Omega)$.

(ii) Interior Regularity: Assume that $\Omega' \subset \Omega$ with $\operatorname{dist}(\partial\Omega', \partial\Omega) > 0$ and that Ω' is smooth. If $\Delta v = f$ in Ω, and $f \in H^r(\Omega)$, then $v \in H^{r+2}(\Omega')$.

Lemma 3.4.20. ***Properties of the Fundamental Solution of Laplace's Equation:*** *Let Ω be a bounded domain. For a source $s \in \Omega$, we define*

$$\Phi_s(x) = \begin{cases} -\dfrac{1}{2\pi} \log|x-s|, & d=2, \\ \dfrac{1}{4\pi} \dfrac{1}{|x-s|}, & d=3. \end{cases} \tag{3.4.27}$$

Then

$$\Delta_x \Phi_s = -\delta_s \tag{3.4.28}$$

$$\Phi_s \in C^\infty(\Omega') \qquad \text{if } s \notin \Omega' \subset \Omega \tag{3.4.29}$$

$$\Phi_s \in L^2(\Omega) \tag{3.4.30}$$

We restate the definition of the specific forward problem that we are interested in here (first introduced in (2.2.13)),

$$u(\boldsymbol{m}, \omega, s) = \mathscr{S}_{\boldsymbol{m},\omega} \delta_s \quad \iff \quad \begin{cases} -(\Delta + \omega^2 m)u &= \delta_s \quad \text{on} \quad \Omega \\ (\partial/\partial n - \mathrm{i}\omega)u &= 0 \quad \text{on} \quad \partial\Omega \end{cases}, \tag{3.4.31}$$

where $m = \sum_k m_k \beta_k$. This is the Helmholtz problem with impedance boundary conditions and a point source at s. We present the following results about u defined by (3.4.31).

Lemma 3.4.21. *If u is the solution to (3.4.31), then there exists $v \in H^2(\Omega)$ such that*

$$u = \Phi_s + v,$$

where Φ_s is defined by (3.4.27).

Proof. Let $v = u - \Phi_s$. By the definition of u and (3.4.28), then

$$\left(\Delta + \omega^2 m\right) v = \delta_s - \left(\delta_s + \omega^2 m \Phi_s\right) = -\omega^2 m \Phi_s, \tag{3.4.32}$$

$$\left(\partial_n - i\omega\right) v = -\left(\partial_n - i\omega\right) \Phi_s. \tag{3.4.33}$$

Since $m \in L^\infty$ and $\Phi_s \in L^2$ by (3.4.30), we have that $m\Phi_s \in L^2$ by Proposition 3.4.14. By (3.4.29), $\left(\partial_n - i\omega\right) \Phi_s \in C^\infty(\partial\Omega) \subset L^2(\partial\Omega)$. Therefore, $v \in H^1(\Omega)$ by Proposition 3.4.18. Rearranging (3.4.32) gives

$$\Delta v = -\omega^2 m v - \omega^2 m \Phi_s,$$

where the first term on the right-hand side is in $L^2(\Omega)$ by Proposition 3.4.14 and the second term is in $L^2(\Omega)$ as shown above. Therefore $v \in H^2(\Omega)$ by Proposition 3.4.19(i). ■

Lemma (3.4.21) shows that the wavefield generated by a point source, defined by (3.4.31), can be split into two parts - the fundamental solution of the Laplace equation, defined in Lemma 3.4.20, and a smooth function in H^2. By Lemma 3.4.21, result (3.4.30) and the fact that $H^2(\Omega) \subset L^2(\Omega)$, we have the following corollary.

Corollary 3.4.22.

$$u \in L^2(\Omega)$$

In the following lemma we examine the smoothness of u in a domain that does not contain the source s. We show that u enjoys higher regularity than shown in Corollary 3.4.22, provided we restrict to a subdomain which does not include the source s.

Lemma 3.4.23. *Suppose the finite element function $\beta_k \in H^r(\Omega)$. For any smooth $\Omega' \subset \Omega$ with $\text{dist}(\partial\Omega', \partial\Omega) > 0$ such that if the source $s \in \Omega \backslash \overline{\Omega'}$, we have $u \in H^{r+2}(\Omega')$.*

Proof. If $s \notin \Omega'$, then there exists a smooth Ω'' such that $\Omega' \subset \Omega''$ with $\text{dist}(\partial\Omega', \partial\Omega'') > 0$ and $s \notin \Omega''$. Since $s \notin \Omega''$, then $\Delta u = -\omega^2 m u$ in Ω''. Since $u \in H^2(\Omega'')$ (by Lemma 3.4.21 and result (3.4.29)), and $m \in H^r(\Omega'')$ (since $\beta_k \in H^r(\Omega'')$), then $mu \in H^r(\Omega'')$ by Proposition 3.4.17. Then $u \in H^{r+2}(\Omega')$ by Proposition 3.4.19(ii). ■

To verify that $\nabla_{\boldsymbol{m}}\phi$ is a smooth enough function of $\boldsymbol{p}$ and $\boldsymbol{m}$, as required by the IFT, we first need to look at the smoothness properties of the wavefield u. We recall that we are considering the two possible choices for the finite elements functions β_k in Proposition 3.4.24. In Corollary 3.4.24, we prove that, in a domain not including the source, when in two dimensions, the wavefield is C^1 with respect to sensor position for both choices of β_k, and when in three-dimensions, the wavefield is C^1 with respect to sensor position for continuous piecewise linear β_k only.

Corollary 3.4.24.

- *If $d = 2$ and β_k are either piecewise constant or continuous piecewise linear, and if $s \notin \Omega'$, then $\dfrac{\partial u}{\partial p_j} \in C(\Omega')$ for $j = 1, \ldots, dN_r$.*
- *If $d = 3$ and β_k are continuous piecewise linear, and if $s \notin \Omega'$, then $\dfrac{\partial u}{\partial p_j} \in C(\Omega')$.*

Proof. We assume that $\beta_k \in H^r(\Omega)$. Then by Lemma 3.4.23, $\partial u/\partial p_j \in H^{r+1}(\Omega')$ for $j = 1, \ldots, dN_r$ if $s \notin \Omega'$. By Proposition 3.4.16, $\partial u/\partial p_j \in C(\Omega')$ if $r + 1 > \frac{d}{2}$. Therefore:

- If $d = 2$, then $r + 1 > \frac{d}{2}$ holds for all $r > 0$. Hence, by Proposition 3.4.15 we get smoothness for both piecewise constant and continuous piecewise linear basis functions.
- If $d = 3$, we require that $r > \frac{1}{2}$, so by Proposition 3.4.15 we need continuous piecewise linear basis functions.

■

We now analyse the differentiability of the wavefield with respect to $\boldsymbol{m}$. In the following lemma we use the multi-index notation $\partial^n_{\boldsymbol{m}}$ to mean $\dfrac{\partial^{|n|}}{\partial^{n_1}_{m_1}\partial^{n_2}_{m_2}\ldots\partial^{n_M}_{m_M}}$, such that $|n| = n_1 + n_2 + \ldots n_M$.

Lemma 3.4.25. *If $\beta_k \in H^r(\Omega)$ for $r > 0$ and $s \notin \Omega'$, for any $n = 1, 2, \ldots$,*

$$\partial^n_{\boldsymbol{m}} u \in H^{r+2}(\Omega').$$

Proof. By differentiating (3.4.31) with respect to m_k, we find that $\partial u/\partial m_k$, for all $k = 1, \ldots, M$, satisfies the PDE,

$$\begin{aligned} -\left(\Delta + \omega^2 m\right)\frac{\partial u}{\partial m_k} &= \omega^2 \beta_k u \qquad \text{in } \Omega \\ (\partial_n - i\omega)\frac{\partial u}{\partial m_k} &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{3.4.34}$$

Since $u \in L^2(\Omega)$ (by Corollary 3.4.22), and $\beta_k \in L^\infty$, we have $\beta_k u \in L^2(\Omega)$ by Proposition 3.4.14. Applying Proposition 3.4.18 gives that $\partial u/\partial m_k \in H^1(\Omega)$. As in the

proof of Lemma 3.4.21, we rearrange the PDE (3.4.34), and use Propsition 3.4.19(i) to get that

$$\frac{\partial u}{\partial m_k} \in H^2(\Omega). \tag{3.4.35}$$

We obtain higher regularity in a smooth subdomain of Ω excluding the source, as follows. Rearranging the PDE in (3.4.34), we have

$$-\Delta\left(\frac{\partial u}{\partial m_k}\right) = \omega^2\left(\beta_k u + m\frac{\partial u}{\partial m_k}\right). \tag{3.4.36}$$

Now we want to apply Proposition 3.4.19 (ii). There exists a smooth subdomain $\Omega'' \supset \Omega'$ with $\text{dist}(\partial\Omega', \partial\Omega'') > 0$ with $s \notin \Omega''$. The right hand side of (3.4.36) is in $H^r(\Omega'')$, by the combination of the following facts: (1) Proposition 3.4.17, (2) $u \in H^2(\Omega'')$ (by Lemma 3.4.21 and result (3.4.29)), (3) $\partial u/\partial m_k \in H^2(\Omega)$ by (3.4.35), (4) β_k and $m \in H^r(\Omega)$. Therefore,

$$\frac{\partial u}{\partial m_k} \in H^{r+2}(\Omega').$$

We have proved the required result for $n = 1$. The PDE and boundary condition (3.4.34) for $n = 2$ is

$$\begin{aligned} -\left(\Delta + \omega^2 m\right)\frac{\partial^2 u}{\partial m_k \partial m_j} &= \omega^2\left(\beta_k \frac{\partial u}{\partial m_j} + \beta_j \frac{\partial u}{\partial m_k}\right) \qquad \text{in } \Omega \\ (\partial_n - i\omega)\frac{\partial^2 u}{\partial m_k \partial m_j} &= 0 \qquad \text{on } \partial\Omega, \end{aligned} \tag{3.4.37}$$

for $k, j = 1, \ldots, M$. Repeating the argument above gives $\partial^2 u/\partial m_k \partial m_j \in H^{r+2}(\Omega')$. By an analogous argument, the result can be extended for arbitrary n. ∎

Corollary 3.4.26. *For $d = 2$ or $d = 3$, and β_k are either piecewise constant and continuous piecewise linear basis functions, and if $s \notin \Omega'$, then for any multi-index n, $\partial^n_{\boldsymbol{m}} u \in C(\Omega')$.*

Proof. We assume $\beta_k \in H^r(\Omega)$ for $r > 0$. Then by Lemma 3.4.25, for any n, $\partial^n_{\boldsymbol{m}} u \in H^{r+2}(\Omega')$ if $s \notin \Omega'$. By Proposition 3.4.16, $\partial^n_{\boldsymbol{m}} u \in C(\Omega')$ if $r + 2 > \frac{d}{2}$. Therefore:

- If $d = 2$, then $r + 2 > \frac{d}{2}$ holds for all $r > 0$.
- If $d = 2$, then $r + 2 > \frac{d}{2}$ also holds for all $r > 0$.

Hence, by Proposition 3.4.15 the result holds for both piecewise constant and continuous piecewise linear basis functions. ∎

To analyse the smoothness of $\nabla_{\boldsymbol{m}}\phi$ with respect to $\boldsymbol{p}$, we also require the following result.

Corollary 3.4.27.

- *If $d = 2$ and β_k are either piecewise constant or continuous piecewise linear, and if $s \notin \Omega'$, then $\dfrac{\partial^2 u}{\partial p_j \partial m_k} \in C(\Omega')$, for $k = 1, \dots M$ and $j = 1, \dots dN_r$.*
- *If $d = 3$ and β_k are continuous piecewise linear functions, and if $s \notin \Omega'$, then $\dfrac{\partial^2 u}{\partial p_j \partial m_k} \in C(\Omega')$, for $k = 1, \dots M$ and $j = 1, \dots dN_r$.*

Proof. We assume that $\beta_k \in H^r(\Omega)$. Then by Lemma 3.4.23 and Lemma 3.4.25, if $s \notin \Omega'$, then $\partial^2 u / \partial p_j \partial m_k \in H^{r+1}(\Omega')$ for $j = 1, \dots, dN_r$ and $k = 1, \dots, M$. By Proposition 3.4.16, $\partial^2 u / \partial p_j \partial m_k \in C(\Omega')$ if $r + 1 > \frac{d}{2}$ holds. Therefore:

- If $d = 2$, then $r + 1 > \frac{d}{2}$ holds for all $r > 0$. By Proposition 3.4.15 we can take piecewise constant or continuous piecewise linear basis functions.
- If $d = 3$, then we require that $r > \frac{1}{2}$. By Proposition 3.4.15 we need continuous piecewise linear basis functions.

■

We now combine the results about the smoothness of the wavefield to prove the smoothness of the lower-level gradient, with respect to sensor position (Theorem 3.4.28) and the model (Theorem 3.4.29). Theorems 3.4.28 and 3.4.29 justify the assumptions made when deriving the upper-level gradient (Assumptions 3.4.1 (ii) and (i) respectively). We note that the proofs for both of these theorems require the assumption that there is no noise added to the synthetic observed data (3.3.4).

Theorem 3.4.28. *In two dimensions, $\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m}, \boldsymbol{p})$ is a C^1 function of $\boldsymbol{p}$ for either choice of basis functions. In three dimensions, $\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m}, \boldsymbol{p})$ is a C^1 function of $\boldsymbol{p}$ when the basis functions are continuous piecewise linear.*

Proof.
In this proof, we assume only one source and one frequency for simplicity. However all

results will hold for the many sources and frequencies. We start by writing (2.3.1) for one source and one frequency, for $k = 1, \ldots, M$ as

$$\frac{\partial \phi}{\partial m_k}(\boldsymbol{m}, \boldsymbol{p}) \;=\; -\Re\left[\left\langle \mathcal{R}(\boldsymbol{p})\frac{\partial u}{\partial m_k}(\boldsymbol{m}), \boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{p})\right\rangle\right].$$

We use the definition of data for the sensor optimisation problem (3.3.4), where we assume no added noise, and the definition of the residual (3.3.5) to write

$$\begin{aligned}\frac{\partial \phi}{\partial m_k}(\boldsymbol{m}, \boldsymbol{p}) \;&=\; -\Re\left[\left\langle \mathcal{R}(\boldsymbol{p})\frac{\partial u}{\partial m_k}(\boldsymbol{m}), \mathcal{R}(\boldsymbol{p})u(\boldsymbol{m}') - \mathcal{R}(\boldsymbol{p})u(\boldsymbol{m})\right\rangle\right] \\ &=\; -\Re\left[\left\langle \frac{\partial u}{\partial m_k}(\boldsymbol{m}; \boldsymbol{p}), u(\boldsymbol{m}'; \boldsymbol{p}) - u(\boldsymbol{m}; \boldsymbol{p})\right\rangle\right], \qquad (3.4.38)\end{aligned}$$

where we write $u(\boldsymbol{m}; \boldsymbol{p})$ to mean the wavefield evaluated at the sensors $\boldsymbol{p}$ (i.e., $\mathcal{R}(\boldsymbol{p})u(\boldsymbol{m}) = u(\boldsymbol{m}; \boldsymbol{p})$). Therefore, to show that $\nabla_{\boldsymbol{m}}\phi$ is continuously differentiable, we must show that $\frac{\partial u}{\partial m_k}(\boldsymbol{m}; \boldsymbol{p})$, $u(\boldsymbol{m}; \boldsymbol{p})$ and $u(\boldsymbol{m}'; \boldsymbol{p})$ are continuously differentiable with respect to $\boldsymbol{p}$. By Corollary 3.4.24, as long as the sensors $\boldsymbol{p}$ are restricted to a domain that does not include the source positions, when $d = 2$, $u(\boldsymbol{m}; \boldsymbol{p})$ and $u(\boldsymbol{m}'; \boldsymbol{p})$ are continuously differentiable with respect to $\boldsymbol{p}$. When $d = 3$, this is true with the additional assumption that β_k are continuous piecewise linear. By Corollary 3.4.27, again when the sensors $\boldsymbol{p}$ are restricted to a domain that does not include the source positions, when $d = 2$, $\frac{\partial u}{\partial m_k}(\boldsymbol{m}; \boldsymbol{p})$ is continuously differentiable with respect to $\boldsymbol{p}$ and when $d = 3$ this is true with the additional assumption that β_k are continuous piecewise linear.
Therefore $\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m}, \boldsymbol{p})$ is continuously differentiable with respect to $\boldsymbol{p}$ in 2D, and is continuously differentiable with respect to $\boldsymbol{p}$ in 3D when the basis functions are continuously piecewise linear. ∎

Theorem 3.4.29. *$\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m}, \boldsymbol{p})$ is a C^1 function of $\boldsymbol{m}$.*

Proof. As before, we assume only one source and one frequency for simplicity in this proof. The proof proceeds as in Theorem 3.4.28 up until (3.4.38), but now we must show that $\frac{\partial u}{\partial m_k}(\boldsymbol{m}; \boldsymbol{p})$, $u(\boldsymbol{m}; \boldsymbol{p})$ and $u(\boldsymbol{m}'; \boldsymbol{p})$ are continuously differentiable with respect to $\boldsymbol{m}$.
By Corollary 3.4.26, $\frac{\partial u}{\partial m_k}(\boldsymbol{m}; \boldsymbol{p})$, $u(\boldsymbol{m}; \boldsymbol{p})$ and $u(\boldsymbol{m}'; \boldsymbol{p})$ are continuously differentiable with respect to $\boldsymbol{m}$, as long as the sensors are restricted to a domain not including the source positions. Therefore $\nabla_{\boldsymbol{m}}\phi$ is at least a C^1 function of $\boldsymbol{m}$ in both 2 and 3 dimensions. ∎

By the Implicit Function Theorem, we have the following corollary.

Corollary 3.4.30. ***Smoothness of $\boldsymbol{m}^{FWI}$ as a function of $\boldsymbol{p}$:***

- *$\boldsymbol{m}^{FWI}$ is a C^1 function of $\boldsymbol{p}$*
- *The partial derivatives of $\boldsymbol{m}^{FWI}$ with respect to sensor positions are given by the matrix products*

$$\frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p})}{\partial p_j} = -\left[H(\boldsymbol{m}^{FWI}(\boldsymbol{p}), \boldsymbol{p})\right]^{-1} \left[\frac{\partial(\nabla_{\boldsymbol{m}}\phi)}{\partial p_j}(\boldsymbol{m}^{FWI}(\boldsymbol{p}), \boldsymbol{p})\right],$$

(which rigorously justifies (3.4.8) used to prove Theorem 3.4.2).

Proof. By Theorems 3.4.28 and 3.4.29, and if the regularisation parameter is chosen large enough, by Theorem 2.4.18, then the assumptions of the Implicit Function Theorem (IFT) (Theorem J-1) hold for

$$\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m}, \boldsymbol{p}) = \boldsymbol{0},$$

where the forward problems are given by (2.2.13). Therefore, the results of the IFT hold, giving the results in Corollary 3.4.30. ∎

We have shown that, as long as the regularisation parameter μ is large enough, $\boldsymbol{m}^{FWI}$ is a smooth function of $\boldsymbol{p}$, and hence the upper-level objective function ψ is a smooth function of $\boldsymbol{p}$. This holds if using either piecewise constant or continuous piecewise linear basis functions when in two dimensions, or continuous piecewise linear basis functions when in three dimensions.

Remark 3.4.31. ***Discrete Version of Smoothness Requirements:*** *In the discrete version of the problem, $u(\boldsymbol{m}; \boldsymbol{p})$, i.e., the wavefield evaluated at the sensor positions, is $R(\boldsymbol{p})\boldsymbol{u}(\boldsymbol{m})$, i.e., the discretised extraction operator, introduced in Remark 2.2.6, applied to the discretised wavefield. We made a comment in Remark 2.2.6 that piecewise linear interpolation of the wavefield is sufficient for FWI, but not for sensor optimisation. This is because $dR(\boldsymbol{p})/dp_j$ in general would not be continuous in this case, and hence by the IFT, the FWI solution would not be a smooth function of sensor positions. Higher-order interpolants are required in implementation of the sensor optimisation problem. This is explained in more detail in Section 5.4.*

3.4.4 Condition for Existence and Uniqueness of Upper-Level Solution

In this section we investigate under what conditions the upper-level problem has a unique solution. Similar to the lower-level problem, we consider how including a convex

regularisation term can ensure a unique solution. Consider the regularised form of the upper-level objective function (3.3.1),

$$\psi(\boldsymbol{p}) = \frac{1}{2N_{m'}} \sum_{\boldsymbol{m}' \in \mathcal{M}'} ||\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')||_2^2 + \frac{\theta}{2}||\boldsymbol{p} - \boldsymbol{p}^*||_2^2 \tag{3.4.39}$$

where $\theta > 0$ is a regularisation parameter, and $\boldsymbol{p}^*$ is some desirable set of sensor coordinates that we encourage $\boldsymbol{p}_{\min}$ to be close to. For instance, $\boldsymbol{p}^*$ could be a set of coordinates that are easier for the seismic survey operators to physically place the sensors at, because they are near the surface for example.

In this analysis we assume that $\boldsymbol{m}^{FWI}$, and hence ψ, is twice continuously differentiable with respect to $\boldsymbol{p}$. The Hessian of (3.4.39) is, for $i, j = 1, \ldots, dNr$,

$$\begin{aligned}
\left(\nabla_{\boldsymbol{p}}^2 \psi(\boldsymbol{p})\right)_{ij} &= \frac{\partial^2 \psi(\boldsymbol{p})}{\partial p_j \partial p_i} \\
&= \sum_{\boldsymbol{m}' \in \mathcal{M}'} \left(\frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')}{\partial p_j}\right)^T \left(\frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')}{\partial p_i}\right) \\
&\qquad - \left(\frac{\partial^2 \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')}{\partial p_j \partial p_i}\right)^T (\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')) + \theta I_{i,j}, \\
&= \sum_{\boldsymbol{m}' \in \mathcal{M}'} \left\langle \frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')}{\partial p_j}, \frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')}{\partial p_i} \right\rangle \\
&\qquad + \left\langle -\frac{\partial^2 \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')}{\partial p_j \partial p_i}, (\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')) \right\rangle + \theta I_{i,j}.
\end{aligned} \tag{3.4.40}$$

The first term on the right-hand side is clearly positive semi-definite. We don't know the properties of the second term from looking at it, so it may be indefinite. The last term is simply a positive constant times the identity matrix. Therefore, we can ensure the Hessian is positive definite overall by making the constant θ large enough.

In the following theorem we derive a lower bound for θ that ensures positive definiteness in the simple case where there is one training model $N_{m'} = 1$ (so we will drop the dependencies on $\boldsymbol{m}'$). We assume that there is only one sensor, and this sensor is being optimised in one coordinate only, so that the sensor-coordinate vector is determined by a single parameter, i.e., $\boldsymbol{p} = p$. We also assume that p lies in a bounded interval (this is true in practice).

Theorem 3.4.32. *Consider the problem of optimising the position of one sensor in one dimension with objective function (3.4.39). Assuming the sensor position lies in a bounded interval, then the upper-level has a unique solution provided θ is sufficiently large.*

Proof. In the one sensor, one training model case, the Hessian of the upper-level (3.4.40)

is just a scalar and can be written as

$$\frac{d^2\psi(p)}{dp^2} = \left\langle \frac{d\boldsymbol{m}^{FWI}(p)}{dp}, \frac{d\boldsymbol{m}^{FWI}(p)}{dp} \right\rangle + \left\langle -\frac{d^2\boldsymbol{m}^{FWI}(p)}{dp^2}, (\boldsymbol{m}' - \boldsymbol{m}^{FWI}(p)) \right\rangle + \theta. \tag{3.4.41}$$

The second term of (3.4.41) is the only part of the Hessian that may not be at least positive semi-definite; let's denote this second term as T_2,

$$T_2(p) = \left\langle -\frac{d^2\boldsymbol{m}^{FWI}(p)}{dp^2}, (\boldsymbol{m}' - \boldsymbol{m}^{FWI}(p)) \right\rangle.$$

The absolute value of T_2 is bounded by

$$|T_2(p)| = \left| \left\langle -\frac{d^2\boldsymbol{m}^{FWI}(p)}{dp^2}, (\boldsymbol{m}' - \boldsymbol{m}^{FWI}(p)) \right\rangle \right| \leq \left\| \frac{d^2\boldsymbol{m}^{FWI}(p)}{dp^2} \right\| \left\| \boldsymbol{m}' - \boldsymbol{m}^{FWI}(p) \right\|. \tag{3.4.42}$$

Since p is assumed to lie in a bounded interval, and $\boldsymbol{m}^{FWI}$ is assumed to be twice continuously differentiable with respect to p, it follows that $|T_2(p)|$ is bounded with respect to p. Therefore, it is possible to choose a regularisation term θ that is larger than this bound, hence ensuring the Hessian is bounded below by a constant for all p. Then by Lemma (H-5) and Theorem (H-6), the minimiser of the sensor placement problem exists and is unique. ∎

3.5 Regularisation Parameter Optimisation

In this section, we consider the optimisation of a lower-level regularisation parameter in addition to the optimisation of sensor positions described earlier. We consider smooth models such that Tikhonov regularisation is appropriate. The lower-level solution will now be written as a function of the Tikhonov regularisation parameter α also, i.e., we write,

$$\boldsymbol{m}^{FWI}(\boldsymbol{p}, \alpha, \boldsymbol{m}') = \underset{\boldsymbol{m}}{\operatorname{argmin}}\ \phi(\boldsymbol{m}, \boldsymbol{p}, \alpha, \boldsymbol{m}'),$$

where the objective function is

$$\phi(\boldsymbol{m}, \boldsymbol{p}, \alpha, \boldsymbol{m}') = \sum_{s\in\mathcal{S}} \sum_{\omega\in\mathcal{W}} \frac{1}{2} ||\boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, s, \omega)||_2^2 + \frac{1}{2}\alpha ||D\boldsymbol{m}||_2^2 + \frac{1}{2}\mu ||\boldsymbol{m}||_2^2. \tag{3.5.1}$$

The gradient of ϕ with respect to the model is, for $k = 1, .., M$,

$$\frac{\partial\phi(\boldsymbol{m}, \boldsymbol{p}, \alpha, \boldsymbol{m}')}{\partial m_k} = -\Re\left\{ \sum_{s\in\mathcal{S}} \sum_{\omega\in\mathcal{W}} \left[\left(\frac{\partial u(\boldsymbol{m}, s, \omega)}{\partial m_k} \right)^* \mathcal{R}(\boldsymbol{p})^* \boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{m}', \boldsymbol{p}, s, \omega) \right] \right\} + \alpha \left(D^T D\boldsymbol{m} \right)_k + \mu m_k. \tag{3.5.2}$$

The solution of the lower-level problem is characterised by the necessary condition

$$\nabla_{\boldsymbol{m}}\phi\left(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha,\boldsymbol{m}'),\boldsymbol{p},\alpha,\boldsymbol{m}'\right)=\boldsymbol{0}\quad\text{for each }\boldsymbol{m}'\in\mathcal{M}'. \tag{3.5.3}$$

The upper-level objective function is now also written as a function of α as follows,

$$\psi(\boldsymbol{p},\alpha)=\frac{1}{2N_{m'}}\sum_{\boldsymbol{m}'\in\mathcal{M}'}||\boldsymbol{m}'-\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha,\boldsymbol{m}')||_2^2, \tag{3.5.4}$$

for each $\boldsymbol{m}'\in\mathcal{M}'$.

3.5.1 Smoothness of of $\boldsymbol{m}^{FWI}$ with respect to the Tikhonov Regularisation Parameter

In this section, we show that the upper-level objective function ψ is a smooth function of the Tikhonov regularisation parameter α by showing that $\boldsymbol{m}^{FWI}$ is a smooth function of α. The result proved here is used in Section 3.5.2, where we derive a formula for the derivative of ψ with respect to α.

We prove that the FWI solution $\boldsymbol{m}^{FWI}$ is a smooth function of α by using the Implicit Function Theorem (IFT). The IFT is written in Appendix J, and similar to Section 3.4.3, we verify the IFT assumptions for the lower-level necessary condition (3.5.3) to show that the IFT conclusions hold for our problem.

The assumptions of the IFT, adapted to our problem are:

- The lower-level gradient $\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m},\boldsymbol{p},\alpha)$ must be continuously differentiable with respect to $\boldsymbol{m}$ (which corresponds to Assumption 3.4.1 (i))
- The lower-level gradient $\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m},\boldsymbol{p},\alpha)$ must be continuously differentiable with respect to α.
- The Hessian at the FWI solution, $H(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha),\boldsymbol{p},\alpha)$, must be invertible (which corresponds to Assumption 3.4.1 (iii)).

In Lemma 3.5.1, we show these assumptions hold for the problem (3.5.3) and hence we can apply the IFT to show that the FWI solution $\boldsymbol{m}^{FWI}$ is continuously differentiable with respect to α. (We note that we suppress the dependence of the notation on $\boldsymbol{m}'$ for this section since only one training model needs to be considered in this discussion.)

Lemma 3.5.1. ***Smoothness of $\boldsymbol{m}^{FWI}$ as a function of α:***

- *$\boldsymbol{m}^{FWI}$ is continuously differentiable with respect to α*
- *The partial derivative of $\boldsymbol{m}^{FWI}$ with respect to α is given by*

$$\frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha)}{\partial \alpha} = -\left[H(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha),\boldsymbol{p},\alpha)\right]^{-1}\left[\frac{\partial(\nabla_{\boldsymbol{m}}\phi)}{\partial\alpha}(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha),\boldsymbol{p},\alpha)\right],$$

where $H \in \mathbb{R}^{M\times M}$ is the Hessian of (3.5.1).

Proof. If the regularisation parameter μ in (3.5.1) is chosen large enough, then by Theorem 2.4.18 the Hessian is positive definite, and hence invertible, for all models, sensor positions and Tikhonov regularisation parameters.

By Theorem 3.4.29, $\nabla_{\boldsymbol{m}}\phi$ is a smooth function of $\boldsymbol{m}$. So it only remains to show that $\nabla_{\boldsymbol{m}}\phi$ is a smooth function of α.

For $k = 1,\ldots,M$, $\nabla_{\boldsymbol{m}}\phi$ is given by (3.5.2), and its derivative with respect to α is

$$\frac{\partial(\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m},\boldsymbol{p},\alpha))}{\partial\alpha} = D^T D\boldsymbol{m}.$$

This derivative exists and is independent of α, and therefore continuous in α. Therefore $\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m},\boldsymbol{p},\alpha)$ is continuously differentiable with respect to α.

Then the assumptions of the Implicit Function Theorem (IFT) (Theorem J-1) hold, giving the result in Corollary 3.5.1. ∎

In conclusion, we have shown that, as long as the regularisation parameter μ is large enough, $\boldsymbol{m}^{FWI}$ is a smooth function of α, and hence the upper-level objective function ψ is a smooth function of α. We note that this holds in both two and three dimension if using either piecewise constant or continuous piecewise linear basis functions.

3.5.2 Gradient of Upper-level Objective Function with respect to the Tikhonov Regularisation Parameter

By the analysis of Section 3.5.1, we have conditions under which ψ is a smooth function of α (namely that the parameter μ is chosen large enough by Theorem 2.4.18). Therefore, under the assumption that this condition is met, we can derive a formula for the derivative of ψ with respect to α.

Theorem 3.5.2. ***Upper-Level Gradient with respect to*** α***:*** *By Lemma 3.5.1, the gradient of the upper-level objective function* ψ (3.5.4) *with respect to the regularisation parameter* α *can be written as*

$$\frac{\partial \psi(\boldsymbol{p},\alpha)}{\partial \alpha} = \frac{1}{N_{m'}} \sum_{\boldsymbol{m}' \in \mathcal{M}'} \left[\left(D^T D \boldsymbol{m}^{FWI} \right)^T H(\boldsymbol{m}^{FWI}, \boldsymbol{p}, \alpha)^{-1} \left(\boldsymbol{m}' - \boldsymbol{m}^{FWI} \right) \right] \tag{3.5.5}$$

where $\boldsymbol{m}^{FWI} = \boldsymbol{m}^{FWI}(\boldsymbol{p}, \alpha, \boldsymbol{m}')$ *and* $H \in \mathbb{R}^{M \times M}$ *is the Hessian of* (3.5.1)*, defined by (2.4.11) with the addition of the term* μI *coming from the convex regularisation and the term* $\alpha D^T D$ *coming from the Tikhonov regularisation.*

Proof. In this proof, we assume only one training model $\boldsymbol{m}'$, i.e., $N_{m'} = 1$, so that we can drop the summation in ψ and all the dependencies on $\boldsymbol{m}'$. We also assume only one source s and one frequency ω and drop the dependencies on these.
The first step is to differentiate ψ (3.5.4) with respect to α to get

$$\frac{\partial \psi(\boldsymbol{p},\alpha)}{\partial \alpha} = \left\langle -\frac{d\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha)}{\partial \alpha}, (\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha)) \right\rangle. \tag{3.5.6}$$

To find an expression for the first term in (3.5.6), $\partial \boldsymbol{m}^{FWI}/\partial \alpha$, we apply the Implicit Function Theorem to the necessary condition (3.5.3). By Lemma 3.5.1, we get

$$\frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha)}{\partial \alpha} = -\left(H(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha), \boldsymbol{p}, \alpha) \right)^{-1} \frac{\partial^2 \phi(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha), \boldsymbol{p}, \alpha)}{\partial \alpha \partial \boldsymbol{m}}, \tag{3.5.7}$$

where H is the Hessian of ϕ with respect to $\boldsymbol{m}$ (which is invertible by Assumption 3.4.1 (iii)). The right-hand side of (3.5.7) is a real $M \times 1$ vector, which, using (3.5.2), is given by,

$$\frac{\partial^2 \phi(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha), \boldsymbol{p}, \alpha)}{\partial \alpha \partial \boldsymbol{m}} = D^T D \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha).$$

Therefore 3.5.7 can be written as

$$\frac{\partial \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha)}{\partial \alpha} = -\left(H(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha), \boldsymbol{p}, \alpha) \right)^{-1} D^T D \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha). \tag{3.5.8}$$

Substituting (3.5.8) into (3.5.6) gives the following,

$$\frac{\partial \psi(\boldsymbol{p},\alpha)}{\partial \alpha} = \left\langle \left(H(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha), \boldsymbol{p}, \alpha) \right)^{-1} D^T D \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha), (\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha)) \right\rangle.$$

Hence, using the symmetry of the Hessian, we get

$$\frac{\partial \psi(\boldsymbol{p},\alpha)}{\partial \alpha} = \left(D^T D \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha) \right)^T H(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha), \boldsymbol{p}, \alpha)^{-1} \left(\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha) \right). \tag{3.5.9}$$

The extension of the result (3.5.9) for many training models is (3.5.5). ∎

Remark 3.5.3. ***Additional Cost of Computing the Gradient:*** *The gradient (3.5.5) can be written in a similar way to that in Equation (3.4.12), i.e.,*

$$\nabla_\alpha \psi(\boldsymbol{p}, \alpha) = \frac{1}{N_{m'}} \sum_{\boldsymbol{m}' \in \mathcal{M}} \left[\boldsymbol{m}^{FWI}(\boldsymbol{p}, \alpha, \boldsymbol{m}')^T D^T D \boldsymbol{\delta}(\boldsymbol{m}^{FWI}(\boldsymbol{p}, \alpha, \boldsymbol{m}'), \boldsymbol{m}') \right]. \quad (3.5.10)$$

where $\boldsymbol{\delta}$ is the solution to (3.4.11). Therefore, when computing the gradient of ψ with respect to α in addition to the gradient of ψ with respect to sensor positions (3.4.12), there are no additional PDE solves. Hence, the Tikhonov regularisation parameter can be optimised along with the sensor coordinates with no extra cost in terms of PDE solves.

3.6 Parameter Optimisation Example

In this section we provide a simple example of sensor placement optimisation, and combined sensor placement and regularisation parameter optimisation as a proof of concept. More details of the implementation are provided in Chapter 5 and more examples are provided in Chapter 6.

Our experiment involves only one training model and three sensors in a transmission setup. We show that our algorithm can optimise the three sensor positions (and regularisation parameter) to improve the quality of the FWI reconstruction of this specific training model. The training model, shown in Figure 3.6.1, is a velocity model with a region of higher wavespeed in the centre, with a maximum of 2100 ms^{-1}, that smoothly decreases in the outward direction. The domain is of size 2500 m $\times$ 2500 m, and is discretised into a 101 $\times$ 101 grid, with a spacing of 25 m. This discretisation results in 10201 model parameters.

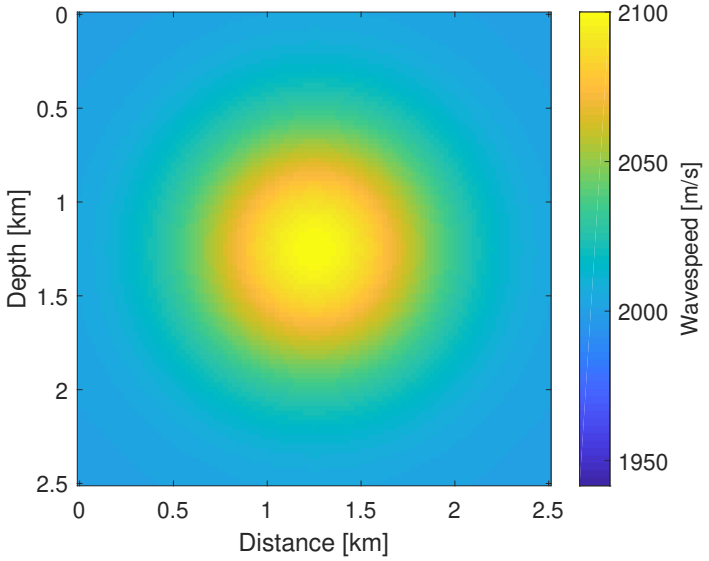
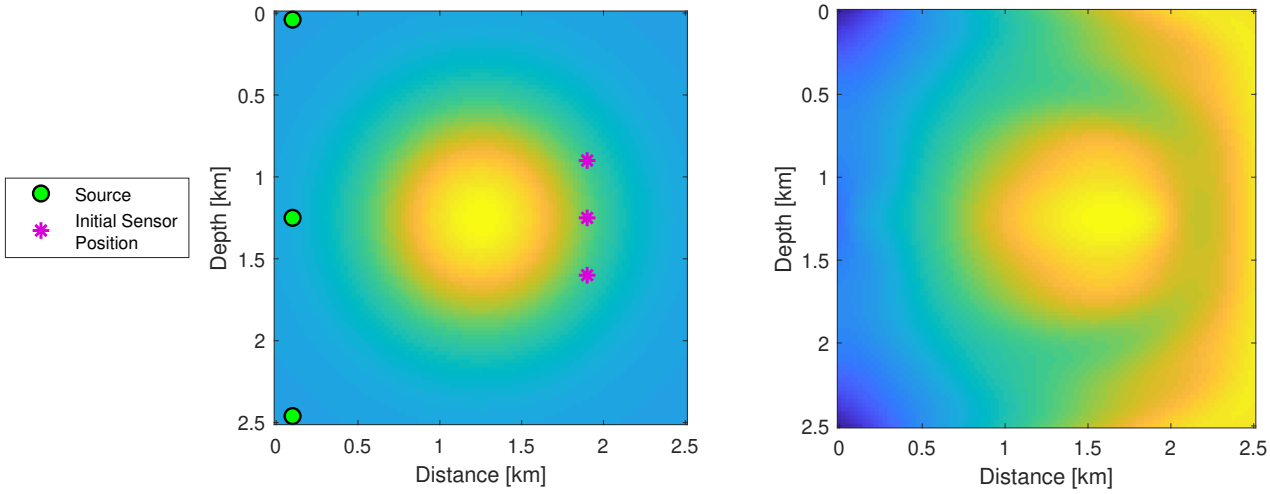


Figure 3.6.1: *Training Model* $\boldsymbol{m}'$

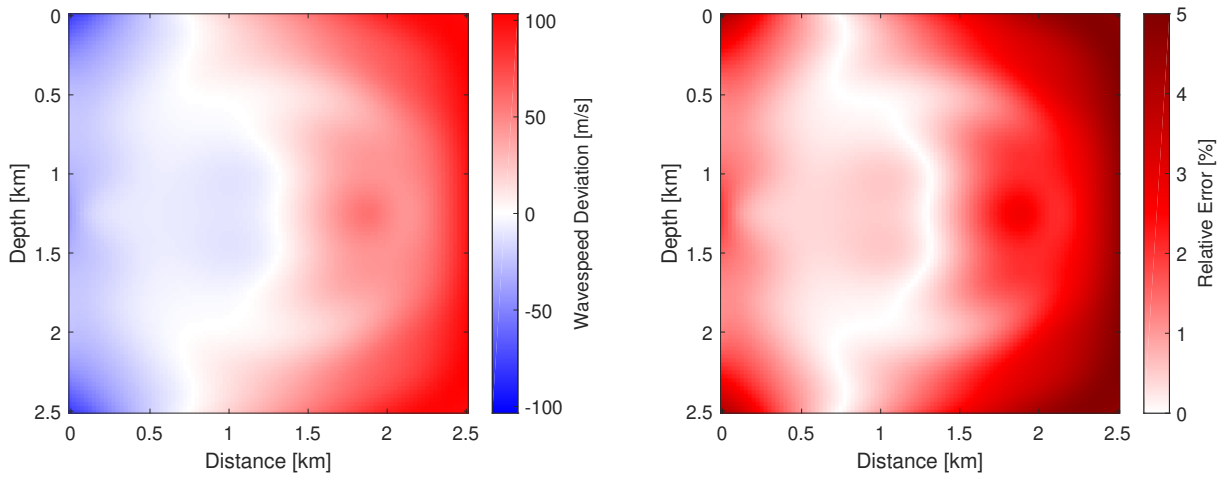
Three sources are placed in a line on the left of the domain. We make an initial guess at the set of sensor positions, $\boldsymbol{p}_0$, and Tikhonov regularisation parameter, which we choose to be $\alpha = 1.25$. The initial setup is overlayed on the training model in Figure 3.6.2 (a), and the resulting FWI reconstruction, corresponding to $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0)$, is shown in Figure 3.6.2 (b). Figure 3.6.2 (b) shows that the initial setup results in a poor reconstruction of the training model. We note that the value of the upper-level objective function (i.e., the FWI error) for this setup is $\psi(\boldsymbol{p}_0) = 3.1590 \times 10^{-4}$. (We make a note about reported values of ψ in Remark 3.6.1).



(a) Initial Setup $\boldsymbol{p}_0$ (b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0)$

Figure 3.6.2: *Initial guess for sensors positions and the resulting reconstruction.*

Figures 3.6.3 (a) and (b) display the errors in the Figure 3.6.2 (b) reconstruction. Figure 3.6.3 (a) shows the absolute error, or wavespeed deviation from the ground truth, given by (2.6.1). This figure demonstrates that the wavespeed being reconstructed on the left side of the domain is lower than the ground truth, and the reconstructed wavespeed on the right side of the domain is larger than the ground truth. Figure 3.6.3 (b) displays the absolute value of the relative percentage error (2.6.2), and shows that the largest errors are on the right-hand side of the sensors and in the corners on the left of the sources. This makes sense since the sensors are recording the waves being produced by the sources on the left and transmitted to the sensors, and so are not getting enough information to correctly reconstruct these parts of the domain during FWI. The maximum relative error is 4.9135% and the average error across the domain is 1.5389%.



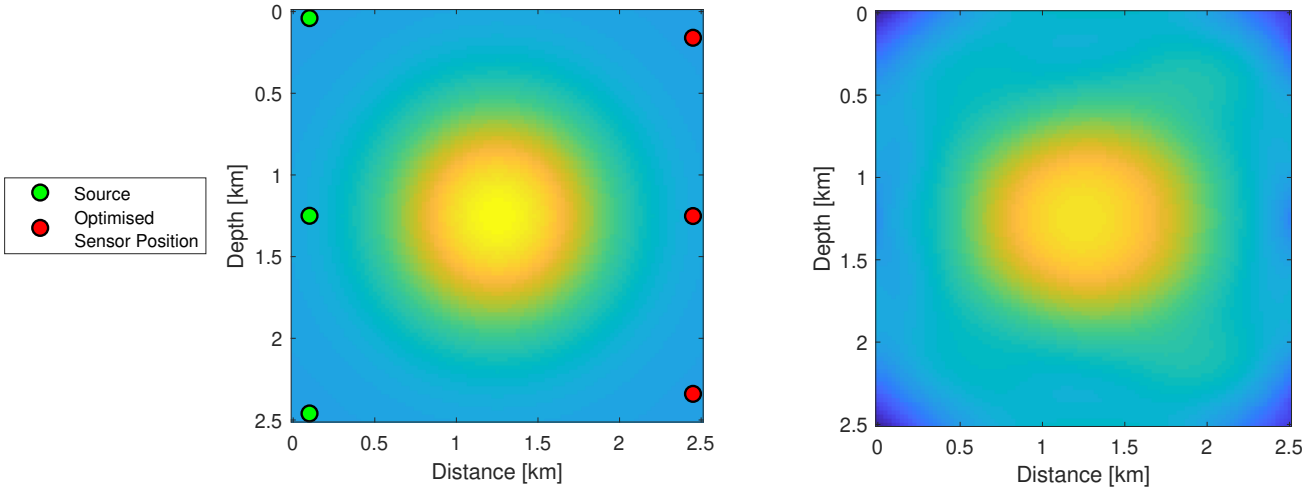
(a) Absolute Error (b) Relative Percentage Error

Figure 3.6.3: *Errors in the reconstruction at the initial guess.*

Optimisation of Sensor Positions: With the starting guess shown in Figure 3.6.2 (a), we use our bilevel algorithm to optimise the sensor positions. We see in Figure 3.6.4 (a) that the sensors have spread out and moved away from the circle of higher wavespeed. These sensor positions make sense as the optimal setup for a transmission experiment since the sensors being spread out from each other and at the opposite side of the domain from the sources allows the sensors to record waves being transmitted through the whole domain. The FWI reconstruction at these optimised sensor positions is shown in Figure 3.6.4 (b), and is a clear improvement on the starting guess reconstruction in Figure 3.6.2 (b). The value of the upper-level objective function at the optimised sensor positions is $\psi(\boldsymbol{p}_{\min}) = 1.3736 \times 10^{-5}$. We define an *improvement factor* to describe the improvement in the quality of our FWI reconstruction image through the use of our bilevel learning algorithm,

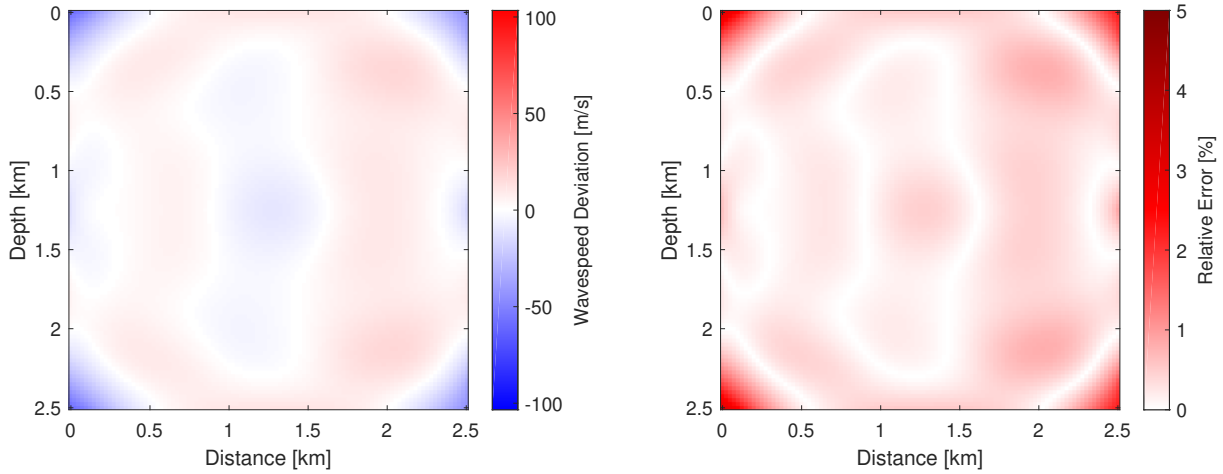
$$\text{Improvement Factor} = \frac{\psi(\boldsymbol{p}_0)}{\psi(\boldsymbol{p}_{\min})}, \tag{3.6.1}$$

i.e., the improvement factor is the ratio between the average FWI error at the starting guess and the average FWI error at at the optimised parameters. For this example, the improvement factor is 22.9983. This means that the FWI error in this example is reduced by a factor of 22.9983 through the use of our algorithm. The reduction in error is evident in Figure 3.6.5, where the error throughout the domain is much less than in Figure 3.6.3. (Note that the Figures 3.6.5 and 3.6.3 are presented on the same scales so that direct comparison can be made). The average relative error throughout the domain is now 0.2935%, and the maximum is 3.0151%. The largest errors occur at the corners.



(a) Optimised Sensor Positions $\boldsymbol{p}_{\min}$ (b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min})$

Figure 3.6.4: *Optimised sensor positions and the resulting reconstruction.*

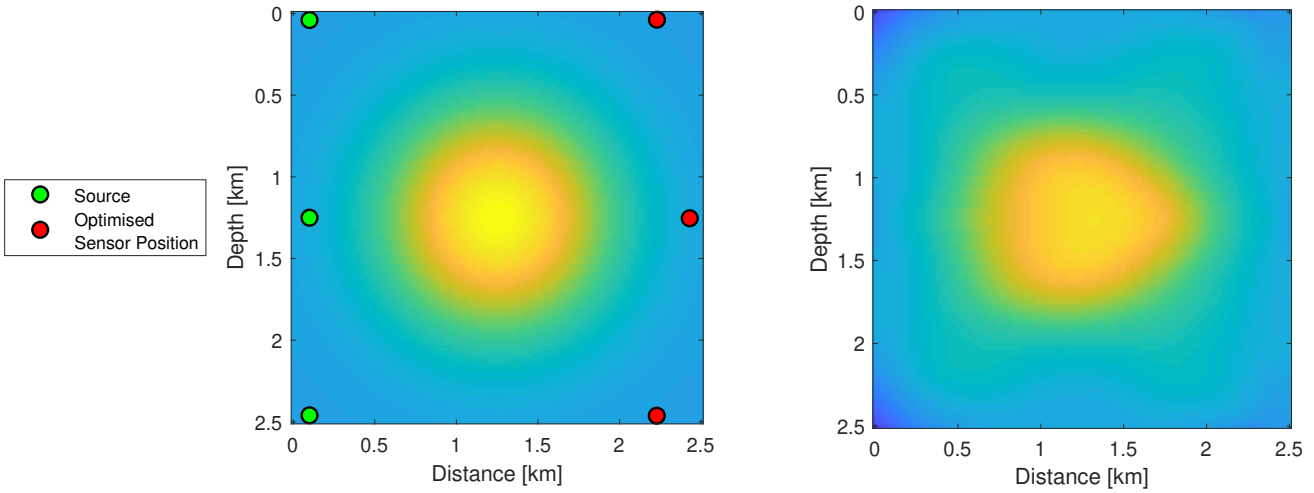


(a) Absolute Error (b) Relative Percentage Error

Figure 3.6.5: *Errors in the reconstruction at the optimised sensor positions.*

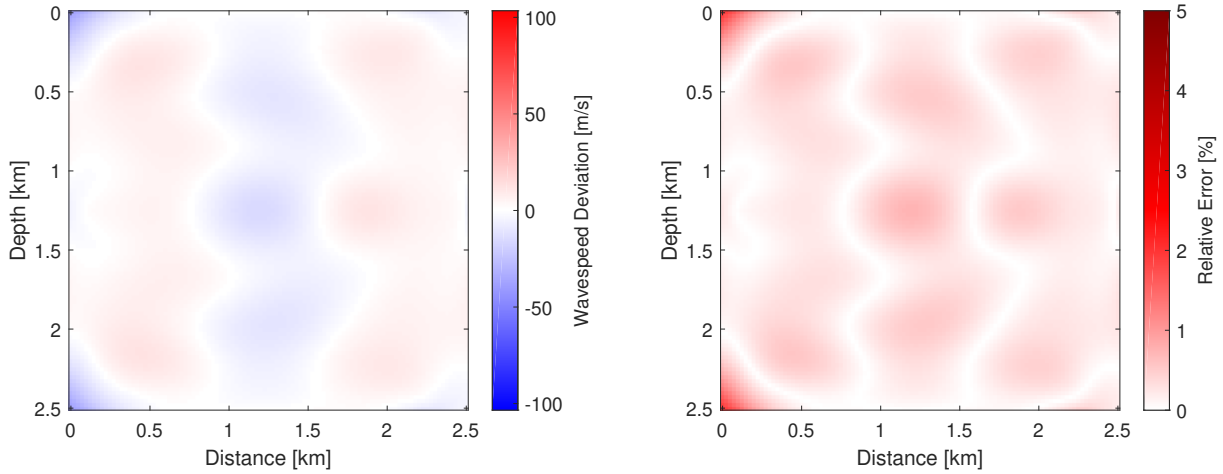
Optimisation of Sensor Positions and Tikhonov Regularisation Parameter: Now we optimise both sensor positions and the Tikhonov regularisation parameter at the same time. The starting guess for sensor positions is again shown in Figure 3.6.2, and the starting guess for the Tikhonov regularisation parameter is $\alpha_0 = 1.25$. The optimised regularisation parameter, $\alpha_{\min}$, is 3.6441, and the optimised sensor positions are shown in Figure 3.6.6 (a) below. Compared to the optimal positions in Figure 3.6.4 (a), the positions here are spread out further from each other and are no longer aligned along the right edge of the domain. Therefore we see that allowing the regularisation parameter to vary during the bilevel algorithm changes the optimal sensor

positions. The value of the upper-level objective function at the optimal parameter is $\psi(\boldsymbol{p}_{\min}, \alpha_{\min}) = 6.6173 \times 10^{-6}$. The improvement factor is now 47.7385, greater than a two-fold improvement over optimising sensor positions only. In Figure 3.6.7, we see that a large improvement in error occurs in the corners, in particular the corners on the right side of the domain where the deviation from the ground truth is now close to zero. The average relative error in the whole domain is 0.2338% and the maximum is 1.9585%, which occurs in the corners on the left.



(a) Optimised Sensor Positions $\boldsymbol{p}_{\min}$ (b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$

Figure 3.6.6: *Optimised sensor positions and Tikhonov regularisation parameter and the resulting reconstruction.*



(a) Absolute Error (b) Relative Percentage Error

Figure 3.6.7: *Errors in the reconstruction at the optimised sensor positions and Tikhonov regularisation parameter.*

Remark 3.6.1. ***Reported Values of ψ:*** *In the experiments in this chapter, and for the rest of this thesis, the reported values of ψ involve a scaling of the grid size squared. For a grid size h, and M model elements, the objective function ψ is the difference between model values on each element, summed up over all elements. For example, in the case where $N_{m'} = 1$, as in the above examples, ψ can be written as*

$$\psi = \frac{1}{2}h^2 \sum_{j}^{M} |m'_j - m_j^{FWI}|^2.$$

This scaling of h^2 accounts for the area of each discretised element. We include this scaling as it allows the comparison of reported ψ values for problems with different grid sizes.

Chapter 4

Symmetry

Chapter Summary: The analysis in this chapter shows that under suitable assumptions, the forward problem (§4.2), the FWI problem (§4.3) and the sensor optimisation problem (§4.4) all have symmetry properties. Under all the required assumptions, we conclude that the optimal set of sensor positions is symmetric. Therefore, for certain problem setups, we can expect symmetry properties in the solution of the sensor placement optimisation problem. In §4.6 we demonstrate some of these properties numerically and show that symmetry can be enforced in the solution of such sensor placement optimisation problems to find an optimal sensor setup with fewer PDE solves. Enforcing a symmetry assumption on the sensor positions can be viewed as a type of regularisation. All analysis in this chapter is novel.

4.1 Introduction

In this section we define the particular problem we are interested in, clarify the notation, define concepts and prove propositions that we need for our symmetry argument.

In this chapter we work in two dimensions only and deal with the following forward problem (Definition 4.1.1). The forward problem is the Helmholtz equation with impedance boundary conditions (first introduced in Section 2.2.2.1).

Definition 4.1.1. ***Forward Problem****: Consider a two-dimensional domain Ω, with Lipschitz boundary $\partial\Omega$ and coordinates $x = (x_1, x_2)$, a model $m = m(x) = \dfrac{1}{c(x)^2}$ (c is the wavespeed), a frequency ω and source term $f = f(x)$. The wavefield $u = u(x)$ satisfies the Helmholtz forward problem*

$$-\left(\Delta + \omega^2 m\right) u = -\left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \omega^2 m\right) u = f \qquad \text{in } \Omega, \tag{4.1.1}$$

with the impedance boundary condition

$$\frac{\partial u}{\partial n} - \mathrm{i}ku = 0 \qquad \text{on } \partial\Omega, \tag{4.1.2}$$

where $k = \omega/c$.

Notation 4.1.2. *We clarify here some notation that may differ from notation in previous chapters. We consider a problem with many sources and many sensors. We denote the position of a source as s, and the set of source positions as $\mathcal{S}$ while sensor positions are written as p and $\mathcal{P}$ denotes the set of sensor positions. The differences in notation, and new notation, are as follows.*

- *We write the source term associated with source position s as f_s.*
- *We write the wavefield that comes from source s, model m and frequency ω as $u_{m,s}(\omega)$. We sometimes suppress the dependence on ω because it is not related to spatial symmetry.*
- *The wavefield measured at a set of sensors is $u_{m,s}(\omega; \mathcal{P}) = \mathcal{R}(\mathcal{P})u_{m,s}(\omega)$. The wavefield evaluated at one sensor is written as $u_{m,s}(\omega; p)$, or when the dependence on ω is suppressed it is just $u_{m,s}(p)$.*
- *The FWI data for a source s, frequency ω and sensor p is denoted $d_s(\omega, p)$. Again we sometimes suppress the dependence on ω.*

Specific Source Term: Here, we consider the source term f to be a delta function, so for any $s \in \mathcal{S}$, $f_s(x) = \delta(x - s) = \delta_s$, where δ_s is defined by

$$\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \delta(x - s)v(x)dx_1dx_2 = v(s), \tag{4.1.3}$$

for any function $v(x)$ smooth enough on $\mathbb{R}^2$. The delta function at $0 = (0, 0)$ will be denoted here by δ_0 or just δ.

The symmetry we study here is about the line $x_2 = 0$. For this reason we define the reflection about the line of symmetry as follows. In Section 4.5 we generalise this definition to handle other symmetries.

Definition 4.1.3. ***Reflection Operation:*** *Let*

$$\mathscr{R} : \mathbb{R}^2 \to \mathbb{R}^2, \ \mathscr{R}(x_1, x_2) = (x_1, -x_2), \tag{4.1.4}$$

i.e., $\mathscr{R}$ is an operator producing a reflection about the x_1-axis.

We denote reflected quantities with a $\widetilde{\ }$, for example, the reflected coordinates are written as $\mathscr{R}(x) = \widetilde{x}$, and the reflected model m is $\widetilde{m}(x) = m(\widetilde{x})$.

Proposition 4.1.4. *$\widetilde{\delta} = \delta$, i.e the delta function is symmetric.*

Proof. We need to show that identity (4.1.3) holds when δ is replaced by $\widetilde{\delta}$, i.e., we show that

$$\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \widetilde{\delta}(x) v(x) dx_1 dx_2 = v(0). \tag{4.1.5}$$

By definition of the reflection operator, $\widetilde{\delta}(x) = \delta(\widetilde{x})$, and so

$$\int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \widetilde{\delta}(x) v(x) dx_1 dx_2 = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \delta(\widetilde{x}) v(x) dx_1 dx_2. \tag{4.1.6}$$

We now make a change of variable in (4.1.6). Let $x = \widetilde{y}$, i.e., $(x_1, x_2) = (y_1, -y_2)$. Therefore, (4.1.6) becomes

$$\begin{aligned} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \delta(\widetilde{x}) v(x) dx_1 dx_2 &= -\int_{+\infty}^{-\infty} \int_{-\infty}^{+\infty} \delta(y) v(\widetilde{y}) dy_1 dy_2 \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \delta(y) \widetilde{v}(y) dy \ = \ \widetilde{v}(0) \ = \ v(0), \end{aligned}$$

where the last two equalities are by the definition of δ and by the definition of the reflection operator respectively. Therefore (4.1.5) holds, and so $\delta = \widetilde{\delta}$. ∎

Proposition 4.1.5. *If $f_s(x) = \delta(x - s)$, then $\widetilde{f}_s(x) = f_{\widetilde{s}}(x)$.*

Proof. First note that $\widetilde{f}_s(x) = f_s(\widetilde{x}) = \delta(\widetilde{x} - s)$ and $f_{\widetilde{s}}(x) = \delta(x - \widetilde{s})$. Therefore to show the result, we need to show that $\delta(\widetilde{x} - s) = \delta(x - \widetilde{s})$.

Proposition 4.1.4 states that $\delta(x) = \delta(\widetilde{x})$, i.e., $\delta(x_1, x_2) = \delta(x_1, -x_2)$. Denoting the coordinates of the source position as $s = (s_1, s_2)$, Proposition 4.1.4 implies

$$\delta(x_1 - s_1, -x_2 - s_2) = \delta(x_1 - s_1, x_2 + s_2) \text{ i.e., } \delta(\widetilde{x} - s) = \delta(x - \widetilde{s}).$$

Therefore the result follows. ∎

Proposition 4.1.6. *For all $x \in \Omega$, and all C^2 functions v, $(\Delta\widetilde{v})(x) = (\Delta v)(\widetilde{x})$.*

Proof. By definition, $\widetilde{v}(x) = v(\widetilde{x})$. By the application of the chain rule,

$$\Delta(\widetilde{v}(x)) = \left(\frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2}\right)(v(x_1, -x_2)) = \left[\left(\frac{\partial^2}{\partial \widehat{x}_1^2} + (-1)^2\frac{\partial^2}{\partial \widehat{x}_2^2}\right)v(\widehat{x}_1, \widehat{x}_2)\right]\Big|_{\widehat{x}_1 = x_1, \widehat{x}_2 = -x_2},$$

and therefore the result holds. ∎

Assumption 4.1.7. *We assume that the domain Ω is symmetric, i.e., if $x \in \partial\Omega$ then $\widetilde{x} \in \partial\Omega$.*

Proposition 4.1.8. *Under Assumption 4.1.7, and denoting the normal vector to $\partial\Omega$ as $n(x)$, we have that*

$$\widetilde{n}(x) = n(\widetilde{x}),$$

i.e., $\mathscr{R}(n(x)) = n(\mathscr{R}(x))$, or $\begin{pmatrix} n_1(x) \\ -n_2(x) \end{pmatrix} = \begin{pmatrix} n_1(\widetilde{x}) \\ n_2(\widetilde{x}) \end{pmatrix}$.

Proof. Let $x \in \partial\Omega$ and let $\partial\Omega_x$ be a small neighbourhood of $\partial\Omega$ surrounding x. We denote the arclength here as s. Then we can write $\partial\Omega_x = \{\gamma(s) : s \in (-\varepsilon, \varepsilon)\}$ for some $\varepsilon > 0$, where $\gamma = (\gamma_1(s), \gamma_2(s))$ is Lipschitz. Arclength parametrisation, implies that $|\gamma'(s)| = 1$. We assume that, for increasing s, $\partial\Omega_x$ is traversed in an anti-clockwise direction.

Then, for $\widetilde{x} = \mathscr{R}(x)$, we write a small neighbourhood of $\partial\Omega$ near $\widetilde{x}$ as $\partial\Omega_{\widetilde{x}} = \{(\gamma_1(s), -\gamma_2(s)) : s \in (-\varepsilon, \varepsilon)\}$. As s increases, $\partial\Omega_{\widetilde{x}}$ is traversed in a clockwise direction.

Therefore, the tangent vector on $\partial\Omega_x$ is $(\gamma_1'(s), \gamma_2'(s))$ and the unit normal is $n(x) = (\gamma_2'(s), -\gamma_1'(s))$. Hence, $\widetilde{n}(x) = (\gamma_2'(s), \gamma_1'(s))$. The tangent vector on $\partial\Omega_{\widetilde{x}}$ is $(\gamma_1'(s), -\gamma_2'(s))$ and the unit normal is $n(\widetilde{x}) = (\gamma_2'(s), \gamma_1'(s))$. The result $\widetilde{n}(x) = n(\widetilde{x})$ follows. ∎

Proposition 4.1.9. *Under Assumption 4.1.7 for all $x \in \partial\Omega$, and all C^1 functions v,*

$$\frac{\partial \widetilde{v}}{\partial n}(x) = \frac{\partial v}{\partial n}(\widetilde{x}).$$

Proof. The normal derivative for a general function $w(x)$ is defined as

$$\frac{\partial w}{\partial n}(x) = \nabla w(x).n(x) = \begin{pmatrix} \dfrac{\partial w(x)}{\partial x_1} \\ \dfrac{\partial w(x)}{\partial x_2} \end{pmatrix} . \begin{pmatrix} n_1(x) \\ n_2(x) \end{pmatrix}. \tag{4.1.7}$$

By definition $\widetilde{v}(x) = v(x_1, -x_2) = v(\widetilde{x})$. The gradient of $\widetilde{v}(x)$ is therefore

$$\nabla \widetilde{v}(x) = \begin{pmatrix} \dfrac{\partial v}{\partial x_1} \\ -\dfrac{\partial v}{\partial x_2} \end{pmatrix} (\widetilde{x})$$

By Proposition 4.1.8, $\mathscr{R}(n(\mathscr{R}(x)) = n(x)$, i.e., $\widetilde{n}(\widetilde{x}) = n(x)$. Therefore, by (4.1.7),

$$\begin{aligned} \frac{\partial \widetilde{v}}{\partial n}(x) &= \begin{pmatrix} \dfrac{\partial v}{\partial x_1}(\widetilde{x}) \\ -\dfrac{\partial v}{\partial x_2}(\widetilde{x}) \end{pmatrix} .n(x) = \begin{pmatrix} \dfrac{\partial v}{\partial x_1}(\widetilde{x}) \\ -\dfrac{\partial v}{\partial x_2}(\widetilde{x}) \end{pmatrix} .\widetilde{n}(\widetilde{x}) = \begin{pmatrix} \dfrac{\partial v}{\partial x_1}(\widetilde{x}) \\ -\dfrac{\partial v}{\partial x_2}(\widetilde{x}) \end{pmatrix} . \begin{pmatrix} n_1(\widetilde{x}) \\ -n_2(\widetilde{x}) \end{pmatrix} \\ &= \nabla v(\widetilde{x}).n(\widetilde{x}) = \frac{\partial v}{\partial n}(\widetilde{x}) \end{aligned}$$

■

4.2 Symmetry of the Forward Problem

This section presents results on the symmetry of the wavefield, i.e., the solution to the forward problem in Definition 4.1.1.

Assumption 4.2.1.

1. *The domain Ω is symmetric, i.e., Assumption 4.1.7 holds.*
2. *The solution to the Boundary Value Problem in Definition 4.1.1 is unique.*

In the following theorem, we show that the wavefield due to a source s and model m is the reflection of the wavefield that comes from the reflected source position $\widetilde{s}$ and

reflected model $\widetilde{m}$. Therefore, if we measure the wavefield from a source s and model m at sensor position p, it will be the same as measuring a wavefield from the reflected source position $\tilde{s}$ and reflected model $\widetilde{m}$ at reflected sensor position $\tilde{p}$.

Theorem 4.2.2. *Under Assumption 4.2.1*

$$\widetilde{u}_{m,s} = u_{\tilde{m},\tilde{s}}, \tag{4.2.1}$$

and thus

$$u_{m,s}(p) = u_{\tilde{m},\tilde{s}}(\tilde{p}). \tag{4.2.2}$$

Proof. The forward problems for the wavefields $u_{m,s}$ and $u_{\widetilde{m},s}$ are respectively,

$$\left(-\Delta - \omega^2 m(x)\right) u_{m,s}(x) = f_s(x) \qquad \text{in } \Omega. \tag{4.2.3}$$

$$\left(-\Delta - \omega^2 \widetilde{m}(x)\right) u_{\tilde{m},s}(x) = f_s(x) \qquad \text{in } \Omega. \tag{4.2.4}$$

It follows from (4.2.4) that the forward problem solved by $u_{\tilde{m},\tilde{s}}$ is

$$\left(-\Delta - \omega^2 \widetilde{m}(x)\right) u_{\tilde{m},\tilde{s}}(x) = f_{\tilde{s}}(x) \qquad \text{in } \Omega. \tag{4.2.5}$$

Now we need to find the PDE that $\widetilde{u}_{m,s}$ solves. First we note that, by Proposition 4.1.6,

$$(\Delta \widetilde{u}_{m,s})(x) = (\Delta u_{m,s})(\widetilde{x}).$$

Then we can write,

$$\begin{aligned} -(\Delta \widetilde{u}_{m,s})(x) - \omega^2 \widetilde{m}(x) \widetilde{u}_{m,s}(x) &= -(\Delta u_{m,s})(\widetilde{x}) - \omega^2 m(\widetilde{x}) u_{m,s}(\widetilde{x}) \\ &= (-\Delta u_{m,s} - \omega^2 m\, u_{m,s})(\widetilde{x}) \\ &= f_s(\widetilde{x}) \ \text{ by } \ (4.2.3) \\ &= f_{\tilde{s}}(x) \ \text{ by Proposition } \ 4.1.5. \end{aligned} \tag{4.2.6}$$

Equations (4.2.5) and (4.2.6) show that $\widetilde{u}_{m,s}$ and $u_{\tilde{m},\tilde{s}}$ solve the same PDE.
Now we need to show that they satisfy the same boundary conditions. By (4.1.2), the boundary condition satisfied by $u_{m,s}$ is

$$\frac{\partial u_{m,s}(x)}{\partial n} - \mathrm{i}k(x) u_{m,s}(x) = 0 \qquad \text{on } \partial\Omega. \tag{4.2.7}$$

This boundary condition holds for all m and s, therefore the boundary condition satisfied by $u_{\tilde{m},\tilde{s}}$ is

$$\frac{\partial u_{\tilde{m},\tilde{s}}(x)}{\partial n} - \mathrm{i}\tilde{k}(x) u_{\tilde{m},\tilde{s}}(x) = 0 \qquad \text{on } \partial\Omega. \tag{4.2.8}$$

Now we need to find the boundary condition that $\widetilde{u}_{m,s}$ satisfies. By Proposition 4.1.9,

$$\frac{\partial \widetilde{u}_{m,s}}{\partial n}(x) = \frac{\partial u_{m,s}}{\partial n}(\widetilde{x}).$$

Then we can write

$$\frac{\partial \widetilde{u}_{m,s}}{\partial n}(x) - \mathrm{i}\widetilde{k}(x)\widetilde{u}_{m,s}(x) = \frac{\partial u_{m,s}}{\partial n}(\widetilde{x}) - \mathrm{i}k(\widetilde{x})u_{m,s}(\widetilde{x}) = \left(\frac{\partial u_{m,s}}{\partial n} - \mathrm{i}ku_{m,s}\right)(\widetilde{x}) = 0 \tag{4.2.9}$$

where the final equality uses (4.2.7). By (4.2.8) and (4.2.9), $\widetilde{u}_{m,s}$ and $u_{\tilde{m},\tilde{s}}$ satisfy the same boundary conditions.
The wavefields are therefore the solution to the same boundary value problem. By uniqueness (Point 2 of Assumption 4.2.1),

$$\widetilde{u}_{m,s} = u_{\tilde{m},\tilde{s}}. \tag{4.2.10}$$

This result may also be written as $u_{m,s} = \widetilde{u}_{\tilde{m},\tilde{s}}$. Evaluating the wavefields at sensor position p gives

$$u_{m,s}(p) = u_{\tilde{m},\tilde{s}}(\widetilde{p}). \tag{4.2.11}$$

■

We now make additional assumptions that allow us to make further conclusions.

Assumption 4.2.3. *$s \in \mathcal{S} \iff \widetilde{s} \in \mathcal{S}$, i.e., the sources are placed symmetrically.*

The following corollary follows directly from Theorem 4.2.2. The corollary states that the set of wavefields that come from the set of source positions $s \in \mathcal{S}$ and model m, is the same as the reflected set of wavefields that come from the reflected set of source positions $\widetilde{s} \in \mathcal{S}$ and reflected model $\widetilde{m}$.

Corollary 4.2.4. *Under Assumptions 4.2.1 and 4.2.3,*

$$\{u_{m,s} : s \in \mathcal{S}\} = \{\widetilde{u}_{\tilde{m},\tilde{s}} : s \in \mathcal{S}\}. \tag{4.2.12}$$

Assumption 4.2.5. *$m = \widetilde{m}$, i.e., the model is symmetric.*

With the additional assumption above in Theorem 4.2.2, we have the following corollary.

Corollary 4.2.6. *Under Assumptions 4.2.1 and 4.2.5,*

$$\widetilde{u}_{m,s} = u_{m,\tilde{s}}$$

and when Assumption 4.2.3 also holds,

$$\{u_{m,s} : s \in \mathcal{S}\} = \{u_{m,\tilde{s}} : s \in \mathcal{S}\} \tag{4.2.13}$$

4.3 Symmetry of the FWI Problem

In this section, we present results on the symmetry of the FWI minimisation problem, using the results of the previous section. We originally defined the FWI Objective function in Definition 2.2.7, however the notation in this chapter is different than in previous chapters so we define the FWI objective function again here.

Definition 4.3.1. ***FWI Objective Function:***
The FWI objective function is

$$\phi(\boldsymbol{m}, \mathcal{S}, \mathcal{P}) = \frac{1}{2} \sum_{\omega \in \mathcal{W}} \sum_{s \in \mathcal{S}} \sum_{p \in \mathcal{P}} ||\varepsilon_{m,s}(\omega, p)||_2^2 + \frac{\mu}{2} ||\boldsymbol{m}||_2^2, \tag{4.3.1}$$

where $\varepsilon_{m,s}(\omega, p) = d_s(\omega, p) - u_{m,s}(\omega; p)$.

Remark 4.3.2. *We note that we have decided to leave the Tikhonov regularisation term out of the FWI objective function in the analysis of this section. However, we note that the derivative matrix in the Tikhonov regularisation term (defined in Remark 2.4.9) is symmetric in the x_1 and x_2 directions. Hence, our results concerning symmetry about the horizontal centre line will still hold with Tikhonov regularisation included. The extension to more general reflections (i.e., those described in Section 4.5) has not been considered.*

Assumption 4.3.3.

1. $s \in \mathcal{S} \iff \tilde{s} \in \mathcal{S}$, *i.e., there is a symmetric layout of sources.*
2. *The FWI data is such that* $\{d_s(p) : s \in \mathcal{S}, p \in \mathcal{P}\} = \{d_{\tilde{s}}(\tilde{p}) : s \in \mathcal{S}, p \in \mathcal{P}\}$

The following theorem tells us that the FWI objective function is symmetric in the model and the set of sensor positions.

Theorem 4.3.4. *When Assumptions 4.2.1 and 4.3.3 hold,*

$$\phi(\boldsymbol{m}, \mathcal{S}, \mathcal{P}) = \phi(\widetilde{\boldsymbol{m}}, \mathcal{S}, \widetilde{\mathcal{P}}) \tag{4.3.2}$$

where $\widetilde{\mathcal{P}} = \{\widetilde{p} : p \in \mathcal{P}\}$.

Proof. We start with the objective function ϕ in Definition 4.3.1, and then use Theorem 4.2.2 result (4.2.2) and Assumptions 4.3.3 (Point 2) to write

$$\begin{aligned}
\phi(\boldsymbol{m}, \mathcal{S}, \mathcal{P}) =& \frac{1}{2} \sum_{\omega \in \mathcal{W}} \sum_{s \in \mathcal{S}} \sum_{p \in \mathcal{P}} ||d_s(\omega, p) - u_{m,s}(\omega; p)||_2^2 + \frac{\mu}{2} ||\boldsymbol{m}||_2^2 \\
=& \frac{1}{2} \sum_{\omega \in \mathcal{W}} \sum_{s \in \mathcal{S}} \sum_{p \in \mathcal{P}} ||d_{\tilde{s}}(\omega, \tilde{p}) - u_{\tilde{m},\tilde{s}}(\omega; \tilde{p})||_2^2 + \frac{\mu}{2} ||\widetilde{\boldsymbol{m}}||_2^2 \\
=& \frac{1}{2} \sum_{\omega \in \mathcal{W}} \sum_{s \in \mathcal{S}} \sum_{p \in \mathcal{P}} ||\varepsilon_{\tilde{m},\tilde{s}}(\omega, \widetilde{p})||_2^2 + \frac{\mu}{2} ||\widetilde{\boldsymbol{m}}||_2^2 \\
=& \frac{1}{2} \sum_{\omega \in \mathcal{W}} \sum_{s \in \mathcal{S}} \sum_{p \in \mathcal{P}} ||\varepsilon_{\tilde{m},s}(\omega, \widetilde{p})||_2^2 + \frac{\mu}{2} ||\widetilde{\boldsymbol{m}}||_2^2 \\
=& \phi(\widetilde{\boldsymbol{m}}, \mathcal{S}, \widetilde{\mathcal{P}})
\end{aligned}$$

where the second last line follows from Assumption 4.3.3 Point 1. ∎

Assumption 4.3.5. *We make the assumption that the solution to the FWI problem is unique.*

Assumption 4.3.5 is in fact guaranteed by Corollary 2.4.20 when μ is chosen by the condition in Theorem 2.4.18.

Theorem 4.3.6. *Under Assumptions 4.2.1, 4.3.3 and 4.3.5,*

$$\boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P}) = \widetilde{\boldsymbol{m}}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}}).$$

Proof. By Theorem 4.3.4,

$$\phi(\boldsymbol{m}, \mathcal{S}, \mathcal{P}) = \phi(\widetilde{\boldsymbol{m}}, \mathcal{S}, \widetilde{\mathcal{P}}).$$

By Assumption 4.3.5, there is a unique global minimum of ϕ, therefore the model $\boldsymbol{m}^{FWI}$ that minimises $\phi(\boldsymbol{m}, \mathcal{S}, \mathcal{P})$ will be equal to the model $\widetilde{\boldsymbol{m}}^{FWI}$ that minimises $\phi(\widetilde{\boldsymbol{m}}, \mathcal{S}, \widetilde{\mathcal{P}})$, i.e.,

$$\boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P}) = \widetilde{\boldsymbol{m}}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}}). \tag{4.3.3}$$

■

Theorem 4.3.6 shows that the FWI solution for a given set of sensor positions is equal to the reflected FWI solution model when the sensors are reflected.

Assumption 4.3.7. *We assume that $p \in \mathcal{P} \iff \tilde{p} \in \mathcal{P}$, i.e., the layout of sensors is symmetric.*

The following corollaries then follow from Theorem 4.3.4. Corollary 4.3.9 shows that, under the stated assumptions, the solution to the FWI problem is symmetric.

Corollary 4.3.8. *Under Assumptions 4.2.1, 4.3.3 and the additional assumption 4.3.7,*

$$\phi(\boldsymbol{m}, \mathcal{S}, \mathcal{P}) = \phi(\widetilde{\boldsymbol{m}}, \mathcal{S}, \mathcal{P}).$$

Corollary 4.3.9. *Under Assumptions 4.2.1, 4.3.3, 4.3.5 and 4.3.7, the result (4.3.3) of Theorem 4.3.6 becomes*

$$\boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P}) = \widetilde{\boldsymbol{m}}^{FWI}(\mathcal{S}, \mathcal{P}).$$

4.4 Symmetry of Sensor Placement Optimisation Problem

We begin by restating the sensor placement upper-level objective function that was introduced in Definition 3.3.2.

Definition 4.4.1. ***Sensor Placement Objective Function:*** *Let $\mathcal{M}'$ be a set of $N_{m'}$ training models, and $\boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P}, \boldsymbol{m}')$ be the solution to the FWI problem for each $\boldsymbol{m}' \in \mathcal{M}'$, with a set of sensors $\mathcal{P}$ and sources $\mathcal{S}$. The sensor placement objective function is*

$$\psi(\mathcal{P}) = \frac{1}{2N_{m'}} \sum_{\boldsymbol{m}' \in \mathcal{M}'} ||\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P}, \boldsymbol{m}')||_2^2. \tag{4.4.1}$$

Assumption 4.4.2.

1. *$s \in \mathcal{S} \iff \tilde{s} \in \mathcal{S}$, i.e., there is a symmetric layout of sources.*
2. *$\boldsymbol{m}' = \widetilde{\boldsymbol{m}}'$, i.e., the training models are symmetric.*
3. *The solution to the FWI problem is unique.*

Assumption 4.4.2 means that the FWI data produced from the training model (i.e., $d_s(p) = u_{m',s}(p)$) satisfies Assumption 4.3.3 Point 2 (by Theorem 4.2.2). Assumption 4.4.2 Point 2 implies that $\boldsymbol{m}' \in \mathcal{M}' \implies \widetilde{\boldsymbol{m}}' \in \mathcal{M}'$.

Theorem 4.4.3 shows that the sensor placement objective function is symmetric in its set of sensor positions.

Theorem 4.4.3. *Under Assumptions 4.2.1 and 4.4.2,*

$$\psi(\mathcal{P}) = \psi(\widetilde{\mathcal{P}}).$$

Proof. By Definition 4.4.1 and Theorem 4.3.6,

$$\begin{aligned}
\psi(\mathcal{P}) &= \frac{1}{2N_{m'}} \sum_{\boldsymbol{m}' \in \mathcal{M}'} ||\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P}, \boldsymbol{m}')||_2^2 \\
&= \frac{1}{2N_{m'}} \sum_{\boldsymbol{m}' \in \mathcal{M}'} ||\boldsymbol{m}' - \widetilde{\boldsymbol{m}}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}}, \boldsymbol{m}')||_2^2 \\
&= \frac{1}{2N_{m'}} \sum_{\widetilde{\boldsymbol{m}}' \in \mathcal{M}'} ||\widetilde{\boldsymbol{m}}' - \widetilde{\boldsymbol{m}}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}}, \widetilde{\boldsymbol{m}}')||_2^2 \\
&= \frac{1}{2N_{m'}} \sum_{\boldsymbol{m}' \in \mathcal{M}'} ||\boldsymbol{m}' - \boldsymbol{m}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}}, \boldsymbol{m}')||_2^2 = \psi(\widetilde{\mathcal{P}}),
\end{aligned} \tag{4.4.2}$$

where the third line uses that the training model is symmetric. ■

Assumption 4.4.4. *There is a unique global minimum of the Sensor Optimisation (i.e., upper-level) problem.*

Assumption 4.4.4 is discussed in Section 3.4.4.

Corollary 4.4.5. *Let $\mathcal{P}_{\min}$ be the solution to the sensor optimisation problem. Under Assumptions 4.2.1, 4.4.2 and 4.4.4,*

$$\psi(\mathcal{P}_{\min}) = \psi(\widetilde{\mathcal{P}}_{\min})$$

and hence

$$\mathcal{P}_{\min} = \widetilde{\mathcal{P}}_{\min}.$$

Corollary 4.4.5 shows that the optimal set of sensor positions is symmetric (under the required assumptions).

4.5 Generalisation of Results

All results in this chapter so far hold for the definition of the reflection operator in Definition 4.1.3, which involves a reflection about the x_1 axis. In this section, we show that these results also hold for more general reflections.

Definition 4.5.1. ***General Reflection Operator:*** *Let*

$$\mathscr{R}(x) = \boldsymbol{R}x,$$

where $\boldsymbol{R}$ is a symmetric real matrix such that $\boldsymbol{R}^2 = I$ and $det(\boldsymbol{R}) = -1$, i.e., $\mathscr{R}$ is the reflection about any line that goes through the origin.

We continue to denote reflected quantities with a $\widetilde{\ }$.

We now prove that the propositions in Section 4.1 (specifically Propositions 4.1.4, 4.1.5, 4.1.6, and 4.1.9) still hold with this more general definition of the reflection operator. When all propositions in Section 4.1 hold, it follows that all results in Sections 4.2, 4.3

and 4.4 also hold.

Proposition 4.5.2. ***General Version of Proposition 4.1.4:***
$\widetilde{\delta} = \delta$, i.e., the delta function is symmetric.

Proof. We need to show that identity (4.1.3) holds when δ is replaced by $\widetilde{\delta}$, i.e.,

$$\int_{\mathbb{R}^2} \widetilde{\delta}(x)v(x)dx = v(0). \tag{4.5.1}$$

By definition, $\widetilde{\delta}(x) = \delta(\widetilde{x})$, and so

$$\int_{\mathbb{R}^2} \widetilde{\delta}(x)v(x)dx = \int_{\mathbb{R}^2} \delta(\widetilde{x})v(x)dx. \tag{4.5.2}$$

We now make a change of variable $x = \widetilde{y}$, i.e., $x = \mathbf{R}y$. Therefore, (4.5.2) becomes

$$\int_{\mathbb{R}^2} \delta(y)v(\widetilde{y})|\det(\mathbf{R})|dy = \int_{\mathbb{R}^2} \delta(y)\widetilde{v}(y)dy = \widetilde{v}(0) = v(0),$$

where we have used that $|\det(\mathbf{R})| = 1$. Therefore (4.5.1) holds and so $\delta(x) = \widetilde{\delta}(x)$. ∎

Proposition 4.5.3. ***General version of Proposition 4.1.5:***
If $f_s(x) = \delta(x - s)$, then,

$$\widetilde{f}_s(x) = f_{\tilde{s}}(x) \tag{4.5.3}$$

Proof. By definition of the reflection operator, we have

$$\widetilde{f}_s(x) = f_s(\widetilde{x}) = \delta(\widetilde{x} - s) = \delta(\mathbf{R}x - s)$$

and

$$f_{\tilde{s}}(x) = \delta(x - \widetilde{s}) = \delta(x - \mathbf{R}s).$$

Therefore, to show (4.5.3), we need to show that $\delta(\mathbf{R}x - s) = \delta(x - \mathbf{R}s)$. This can be shown with the result of Proposition 4.5.2, which states that $\delta(x) = \delta(\widetilde{x})$, i.e., $\delta(x) = \delta(\mathbf{R}x)$. Using this, and the fact that $\mathbf{R}^2 = I$, gives that

$$\delta(\mathbf{R}x - s) = \delta(\mathbf{R}(x - \mathbf{R}s)) = \delta(x - \mathbf{R}s),$$

i.e., $\delta(\widetilde{x} - s) = \delta(x - \widetilde{s})$, and so $\widetilde{f}_s(x) = f_{\tilde{s}}(x)$. ∎

Proposition 4.5.4. ***General version of Proposition 4.1.6:*** *For all $x \in \Omega$, and all C^2 functions v,*

$$(\Delta\widetilde{v})(x) = (\Delta v)(\widetilde{x}), \tag{4.5.4}$$

Proof. We first write the chain rule for a general function $f : \mathbb{R}^d \to \mathbb{R}$ and $g : \mathbb{R}^d \to \mathbb{R}^d$,

$$\frac{\partial}{\partial x_i}(f(g(x)) = \sum_k \left(\left(\frac{\partial f}{\partial x_k}\right)(g(x))\right)\left(\frac{\partial g_k}{\partial x_i}(x)\right).$$

Applying the chain rule to our problem, where $\widetilde{v}(x) = v(\widetilde{x}) = v(\mathbf{R}x)$,

$$\frac{\partial}{\partial x_i}(v(\mathbf{R}x)) = \sum_k \left(\left(\frac{\partial v}{\partial x_k}\right)(\mathbf{R}x)\right)\left(\frac{\partial(\mathbf{R}x)_k}{\partial x_i}\right) \tag{4.5.5}$$

Denoting the (k,m)th entry of $\mathbf{R}$ as r_{km}, we can write,

$$(\mathbf{R}x)_k = \sum_m r_{km}x_m$$

and

$$\frac{\partial(\mathbf{R}x)_k}{\partial x_i} = \sum_m r_{km}\delta_{mi} = r_{ki}.$$

Using these identities in (4.5.5), we get

$$\begin{aligned}\frac{\partial}{\partial x_i}(v(\mathbf{R}x)) &= \sum_k \left(\left(\frac{\partial v}{\partial x_k}\right)(\mathbf{R}x)\right) r_{ki} = \sum_k r_{ki}\left(\left(\frac{\partial v}{\partial x_k}\right)(\mathbf{R}x)\right) \\ &= \left(\mathbf{R}^T\nabla v(\mathbf{R}x)\right)_i\end{aligned} \tag{4.5.6}$$

and so,

$$\nabla(v(\mathbf{R}x)) = \mathbf{R}^T((\nabla v)(\mathbf{R}x)). \tag{4.5.7}$$

To find $\Delta(v(\mathbf{R}x))$, we take another derivative of (4.5.7), which gives

$$\begin{aligned}\Delta(v(\mathbf{R}x)) = \nabla.(\nabla(v(\mathbf{R}x))) &= \sum_i \frac{\partial}{\partial x_i}\left(\frac{\partial}{\partial x_i}(v(\mathbf{R}x))\right) \\ &= \sum_i \frac{\partial}{\partial x_i}\left(\sum_k r_{ki}\left(\left(\frac{\partial v}{\partial x_k}\right)(\mathbf{R}x)\right)\right), \\ &= \sum_i\sum_k r_{ki}\frac{\partial}{\partial x_i}\left(\left(\frac{\partial v}{\partial x_k}\right)(\mathbf{R}x)\right),\end{aligned} \tag{4.5.8}$$

where the second last equality comes from (4.5.6). We apply the chain rule again to explicitly find the derivative term appearing in (4.5.8),

$$\frac{\partial}{\partial x_i}\left(\left(\frac{\partial v}{\partial x_k}(\mathbf{R}x)\right)\right) = \sum_l \left(\frac{\partial^2 v}{\partial x_l \partial x_k}\right)(\mathbf{R}x)\frac{\partial(\mathbf{R}x)_l}{\partial x_i}$$
$$= \sum_l \left(\frac{\partial^2 v}{\partial x_l \partial x_k}\right)(\mathbf{R}x) r_{li}. \tag{4.5.9}$$

Substituting (4.5.9) back into (4.5.8) gives

$$\Delta(v(\mathbf{R}x)) = \sum_i \sum_k \sum_l r_{ki} r_{li} \left(\frac{\partial^2 v}{\partial x_k \partial x_l}\right)(\mathbf{R}x) = \sum_k \sum_l \left(\sum_i r_{ki} r_{li}\right)\left(\frac{\partial^2 v}{\partial x_k \partial x_l}\right)(\mathbf{R}x)$$
$$= \sum_k \sum_l \delta_{lk} \left(\frac{\partial^2 v}{\partial x_k \partial x_l}\right)(\mathbf{R}x) = \sum_k \left(\frac{\partial^2 v}{\partial x_k \partial x_k}\right)(\mathbf{R}x) = (\Delta v)(\mathbf{R}x),$$

where $\sum_i r_{ki} r_{li} = \delta_{lk}$ because $\mathbf{R}^T\mathbf{R} = I$ (since $\mathbf{R}^2 = I$ and $\mathbf{R} = \mathbf{R}^T$ by Definition 4.5.1). ■

Proposition 4.5.5. ***General version of Proposition 4.1.8:***
Under Assumption 4.1.7, and denoting the normal vector to $\partial\Omega$ as $n(x)$, we have that

$$\widetilde{n}(x) = n(\widetilde{x}),$$

i.e., $\mathscr{R}(n(x)) = n(\mathscr{R}(x))$, or $\boldsymbol{R}n(x) = n(\boldsymbol{R}(x))$.

Proof. Following the same steps as in the proof of Proposition 4.1.8, let $x \in \partial\Omega$ and let $\partial\Omega_x$ be a small neighbourhood of $\partial\Omega$ surrounding x. Then we can write $\partial\Omega_x = \{\gamma(s) : s \in (-\varepsilon, \varepsilon)\}$ for some $\varepsilon > 0$, where $\gamma = (\gamma_1(s), \gamma_2(s))$ is Lipschitz. Using arclength parametrisation, $|\gamma'(s)| = 1$. We assume that, for increasing s, $\partial\Omega_x$ is traversed in an anti-clockwise direction. The tangent vector on $\partial\Omega_x$ is $(\gamma_1'(s), \gamma_2'(s))$ and the unit normal is therefore $n(x) = (\gamma_2'(s), -\gamma_1'(s))$, or equivalently,

$$n(x) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\gamma'(s).$$

Then, for $\widetilde{x} = \mathbf{R}x$, we write a small neighbourhood of $\partial\Omega$ near $\widetilde{x}$ as $\partial\Omega_{\widetilde{x}} = \{\widetilde{\gamma} : s \in (-\varepsilon, \varepsilon)\} = \{\mathbf{R}\gamma : s \in (-\varepsilon, \varepsilon)\}$. As s increases, $\partial\Omega_{\widetilde{x}}$ is traversed in a clockwise direction. The tangent vector on $\partial\Omega_{\widetilde{x}}$ is $\mathbf{R}\gamma'$ and the unit normal is

$$n(\widetilde{x}) = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\mathbf{R}\gamma'(s).$$

We must show that $n(\widetilde{x}) = \widetilde{n}(x)$, i.e.,

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}\mathbf{R}\gamma'(s) = \mathbf{R}\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\gamma'(s),$$

or equivalently

$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \mathbf{R} = \mathbf{R} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

We begin by multiplying by the inverse of the first matrix on the left, giving the following as what we need to show,

$$\mathbf{R} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \mathbf{R} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

or equivalently, writing out the components of $\mathbf{R}$,

$$\begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

i.e.,

$$\begin{pmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{pmatrix} = \begin{pmatrix} -r_{22} & r_{21} \\ r_{12} & -r_{11} \end{pmatrix} \tag{4.5.10}$$

We can show that (4.5.10) is true by noting the properties of the reflection matrix $\mathbf{R}$ (stated in Definition 4.5.1). The right hand side of (4.5.10) is $\mathbf{R}^{-1}$ by Cramer's rule and the fact that $\det(\mathbf{R}) = -1$. Therefore, (4.5.10) simply states that $\mathbf{R} = \mathbf{R}^{-1}$, or equivalently $\mathbf{R}^2 = I$, which is true by definition. Therefore we have shown that $n(\widetilde{x}) = \widetilde{n}(x)$. ∎

Proposition 4.5.6. ***General version of 4.1.9:***
Under Assumption 4.1.7, for all $x \in \partial\Omega$ and all C^1 functions v,

$$\frac{\partial \widetilde{v}}{\partial n}(x) = \frac{\partial v}{\partial n}(\widetilde{x}).$$

Proof. The gradient of $\widetilde{v}(x)$ satisfies (by (4.5.7))

$$\nabla v(\widetilde{x}) = \mathbf{R}^T (\nabla v)(\widetilde{x})$$

By Proposition 4.5.5, $\mathscr{R}(n(\mathscr{R}(x)) = n(x)$, i.e., $\widetilde{n}(\widetilde{x}) = n(x)$. Therefore,

$$\begin{aligned} \frac{\partial \widetilde{v}}{\partial n}(x) &= \left(\mathbf{R}^T (\nabla v)(\widetilde{x})\right).n(x) = \left(\mathbf{R}^T (\nabla v)(\widetilde{x})\right).\widetilde{n}(\widetilde{x}) = \left(\mathbf{R}^T (\nabla v)(\widetilde{x})\right).(\mathbf{R}n(\widetilde{x})) \\ &= \left((\nabla v)(\widetilde{x})\right).(\mathbf{R}^2 n(\widetilde{x})\right) = ((\nabla v)(\widetilde{x})).n(\widetilde{x}) = \frac{\partial v}{\partial n}(\widetilde{x}) \end{aligned}$$

where the second last equality uses the fact that $\mathbf{R}^2 = I$. ∎

We have shown in this section that Propositions 4.1.4, 4.1.5, 4.1.6, and 4.1.9 hold for the general definition of reflection (Definition 4.5.1). Therefore, under the required assumptions, all the results in Sections 4.2, 4.3 and 4.4 hold for any reflection about the origin, and, most importantly, the optimal set of sensor positions is symmetric about any axis through the origin.

4.6 Experiments

This section numerically demonstrates some of the main results from this chapter - Theorem 4.2.2, Theorem 4.3.6 and Corollary 4.4.5 - and how the result of Corollary 4.4.5 can be exploited in our bilevel sensor optimisation algorithm.

Each of these results involve assumptions. We enforce assumptions involving symmetry, however, we cannot numerically guarantee assumptions about uniqueness, i.e., that the forward problem has a unique solution (Assumption 4.2.1), or that the sensor optimisation problem has a unique global minimum (Assumption 4.4.4). We include a convex regularisation term on the lower-level with the purpose of making sure that FWI has a unique solution (Assumptions 4.3.5 and 4.4.2 Point 3), but we cannot numerically ensure that the lower-level solution is in fact unique, because we do not know explicitly what choice of regularisation parameter μ guarantees this. The experiments in this section show that these uniqueness assumptions are sufficient but not necessary for the theoretical results of this chapter to hold in practice.

Experiment 1: Demonstrating Theorem 4.2.2
Theorem 4.2.2 states that, assuming the domain is symmetric and that the solution to the forward problem (Definition 4.1.1) is unique, then the wavefield due to a source position s and model m is equal to the reflection of the wavefield due to the reflected source position $\widetilde{s}$ and reflected model $\widetilde{m}$. We illustrate this result by visualising numerical solutions of the discretised forward problem for a specific example, i.e., we show that $\widetilde{\mathbf{u}}_{m,s} = \mathbf{u}_{\widetilde{m},\widetilde{s}}$ in practice. The discretised model $\boldsymbol{m}$ and source position s that is used in this experiment are shown in Figure 4.6.1. The corresponding solution to the Helmholtz equation with model $\boldsymbol{m}$ and point source at position s is shown in Figure 4.6.2. Since the solution to the Helmholtz equation is complex, we visualise the real and imaginary parts of the wavefield separately.

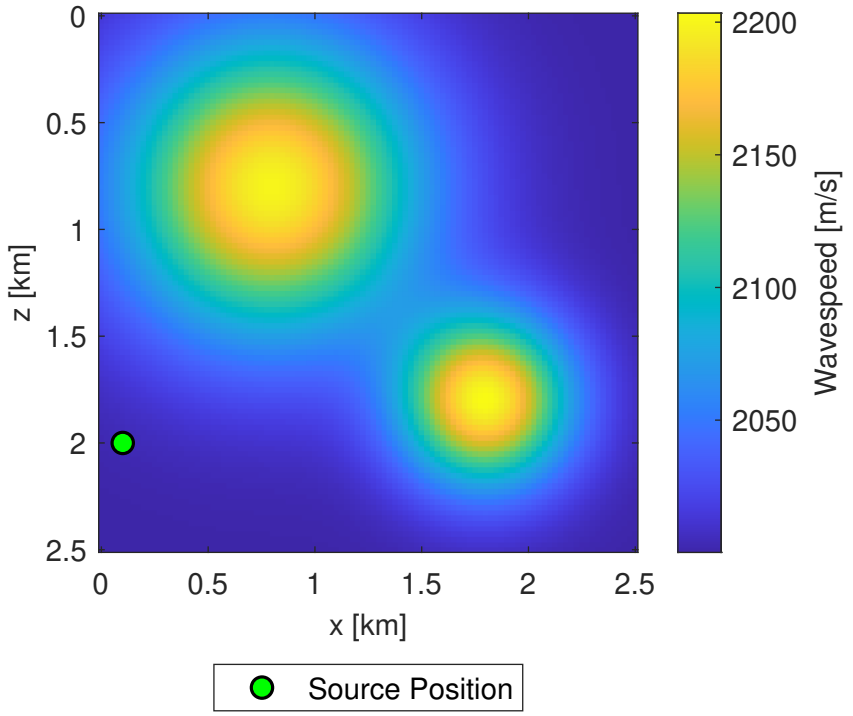
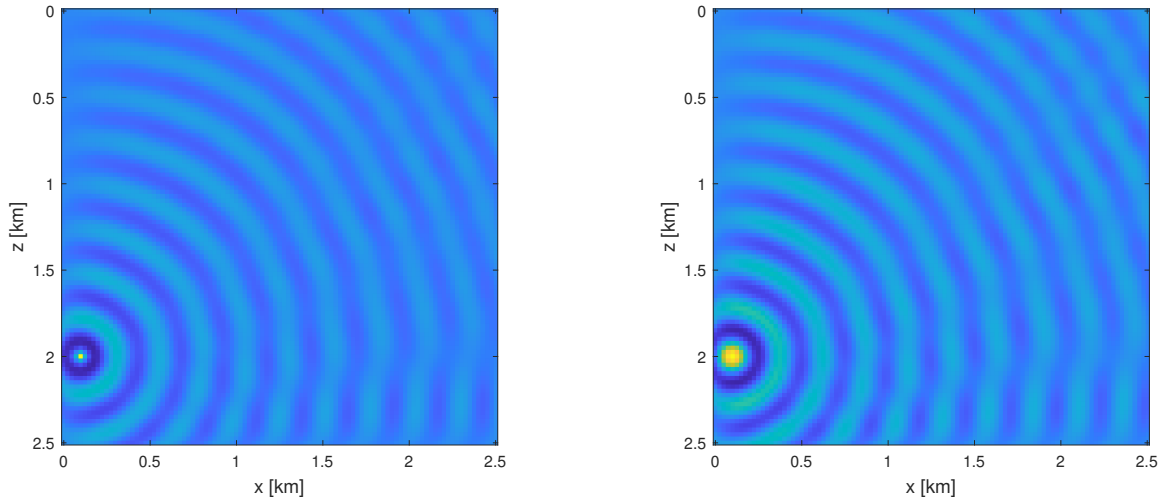


Figure 4.6.1: *The model* $\boldsymbol{m}$ *and source position* s *used to produce the wavefield* $\boldsymbol{u}_{\boldsymbol{m},s}$.



(a) Real Part (b) Imaginary Part

Figure 4.6.2: *Wavefield* $\boldsymbol{u}_{\boldsymbol{m},s}$

We choose a line of reflection at $z = 1.25$ km, shown in Figure 4.6.3 for clarity, and reflect the computed wavefield in Figure 4.6.2 about this line to give the reflected wavefield $\widetilde{\mathbf{u}}_{\boldsymbol{m},s}$ shown in Figure 4.6.4.

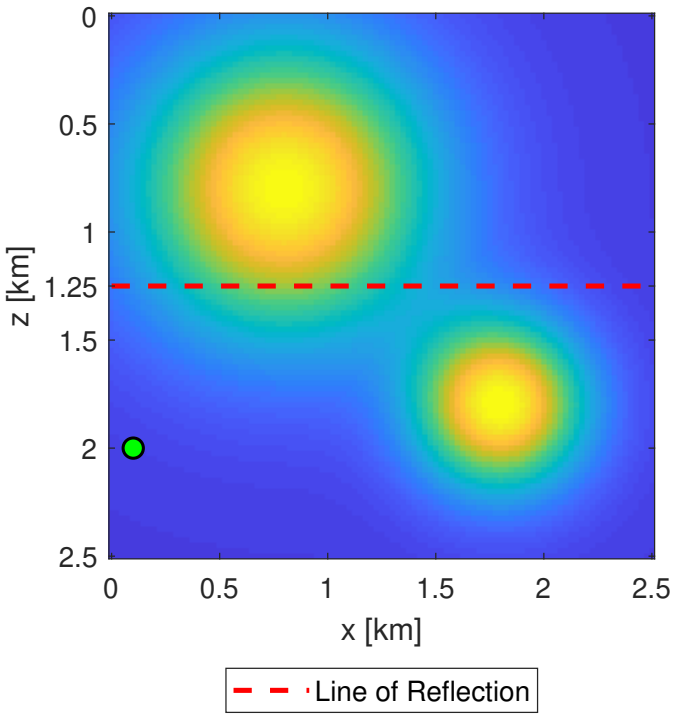
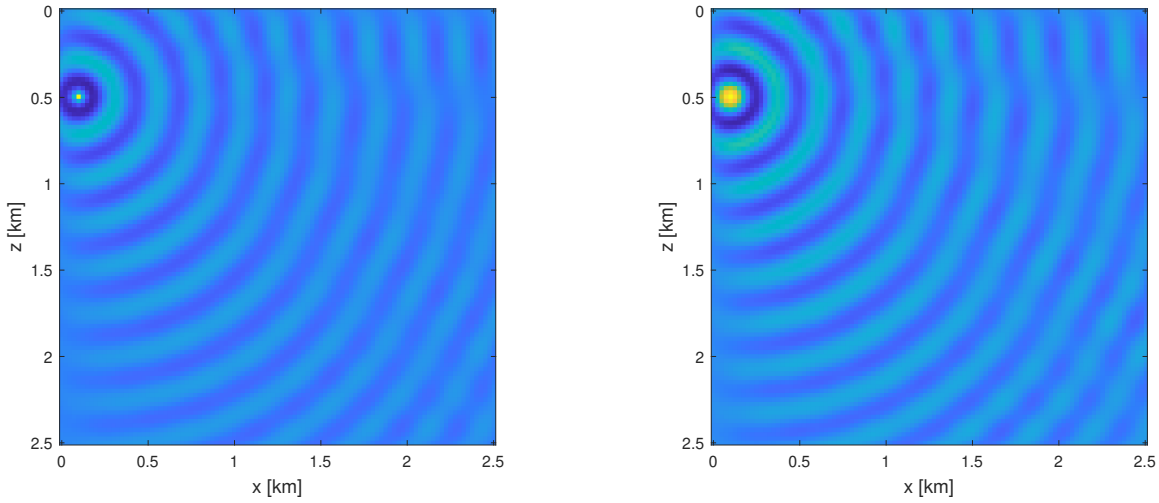


Figure 4.6.3: *The line of relection.*



(a) Real Part (b) Imaginary Part

Figure 4.6.4: *Reflected wavefield* $\widetilde{\boldsymbol{u}}_{\boldsymbol{m},s}$

We then reflect the model and source position about the line of reflection shown in Figure 4.6.3. The resulting reflected model $\widetilde{\boldsymbol{m}}$ and reflected source position $\widetilde{s}$ are shown in Figure 4.6.5. The corresponding solution to the Helmholtz equation with model $\widetilde{\boldsymbol{m}}$ and point source at position $\widetilde{s}$, i.e., $\mathbf{u}_{\widetilde{m},\widetilde{s}}$, is shown in Figure 4.6.6.

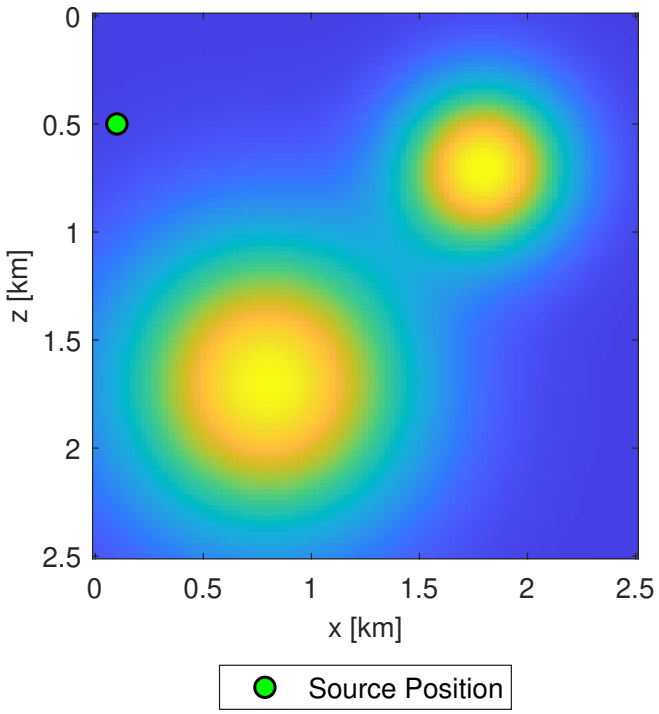
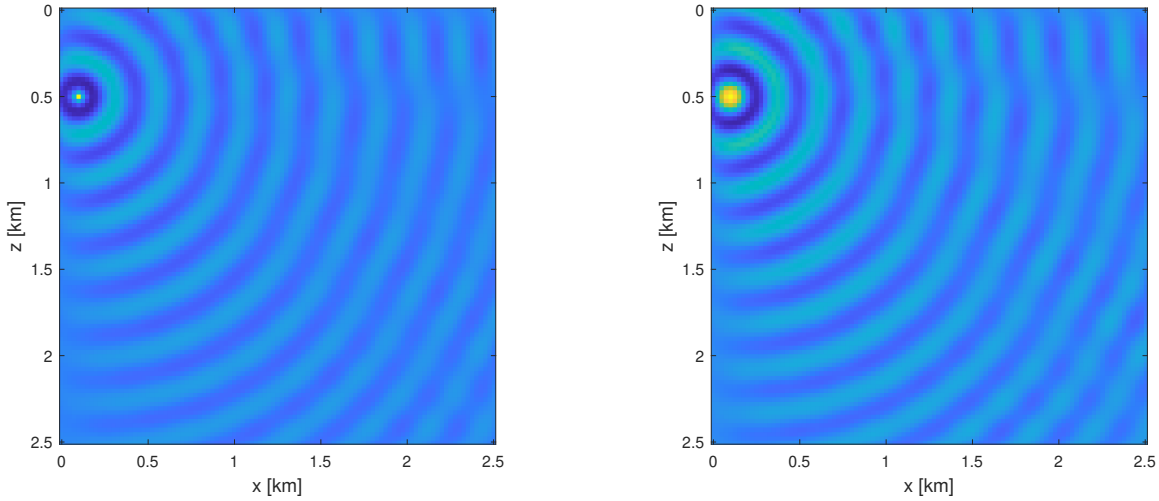


Figure 4.6.5: *The reflected model* $\widetilde{\boldsymbol{m}}$ *and reflected source position* $\tilde{s}$.



(a) Real Part (b) Imaginary Part

Figure 4.6.6: *Wavefield from reflected model and reflected source* $\boldsymbol{u}_{\tilde{m},\tilde{s}}$

Visually, the wavefields in Figures 4.6.4 and 4.6.6 are identical. Computationally, the infinity norm of the absolute value of difference between $\widetilde{\mathbf{u}}_{\boldsymbol{m},s}$ and $\mathbf{u}_{\tilde{m},\tilde{s}}$ is 2.0490 $\times 10^{-16}$, which is smaller than machine precision, meaning that the difference between the wavefields is numerically zero at every point. Therefore, this experiment numerically demonstrates the result of Theorem 4.2.2 for the discretised problem, i.e.,

$$\widetilde{\mathbf{u}}_{\boldsymbol{m},s} = \mathbf{u}_{\tilde{m},\tilde{s}}.$$

Experiment 2: Demonstrating Theorem 4.3.6
We demonstrate numerically the result of Theorem 4.3.6, which states that the FWI solution for a given set of sensors is equal to the reflected FWI solution when the sensors are reflected. We ensure the following assumptions are true:

- The domain is symmetric (Assumption 4.2.1 Point 1).
- The set of source positions $\mathcal{S}$ is symmetric (Assumption 4.3.3 Point 2).
- The data has symmetric properties (from Assumption 4.3.3 Point 2) by generating synthetic data from a symmetric model.

The velocity model and source positions are shown in Figure 4.6.7, which are symmetric about the line of reflection at $z = 1.25$ km.

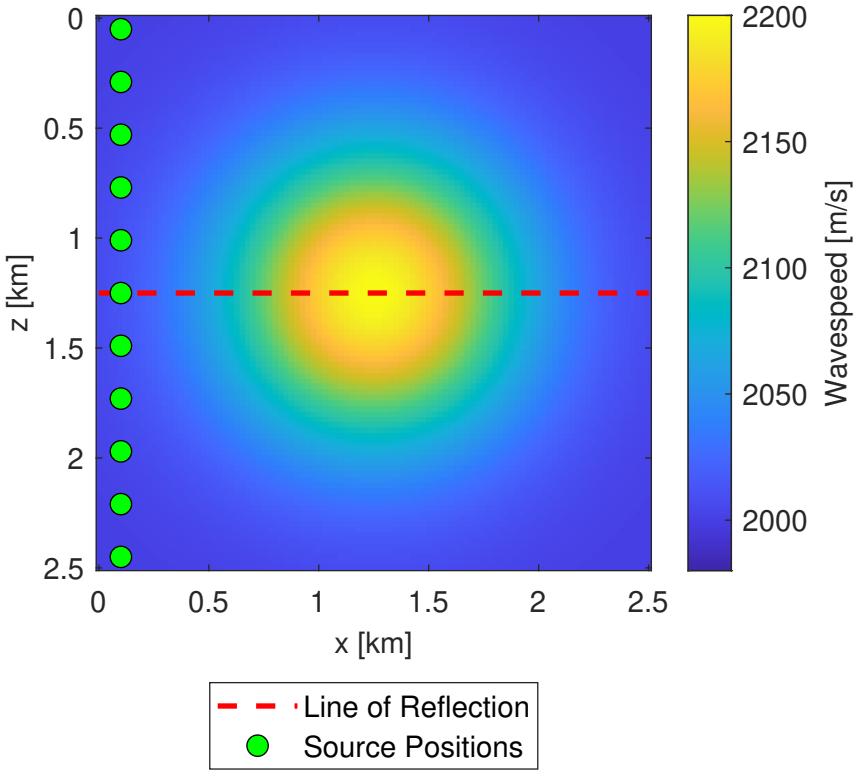
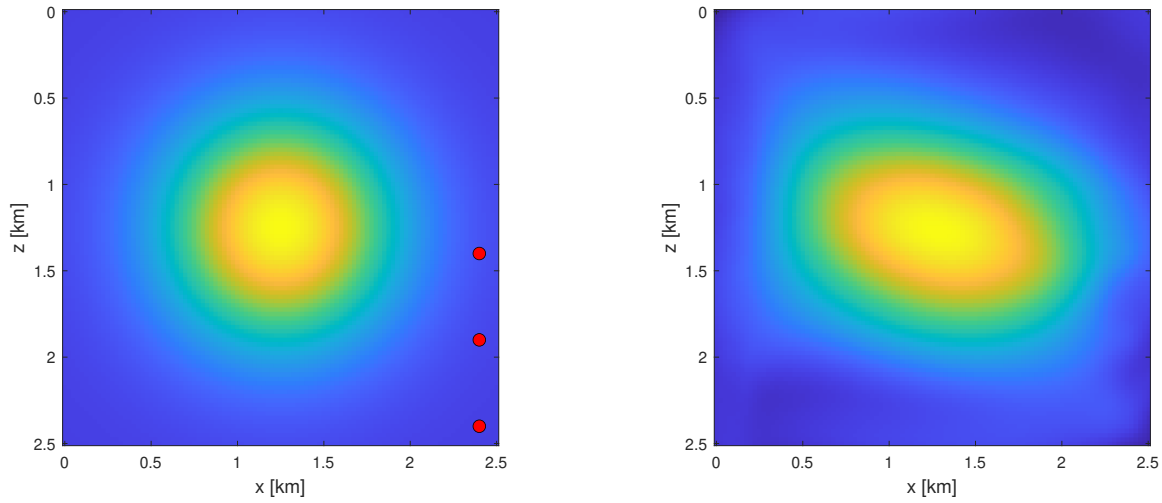


Figure 4.6.7: *Symmetric ground truth model and source positions used for FWI.*

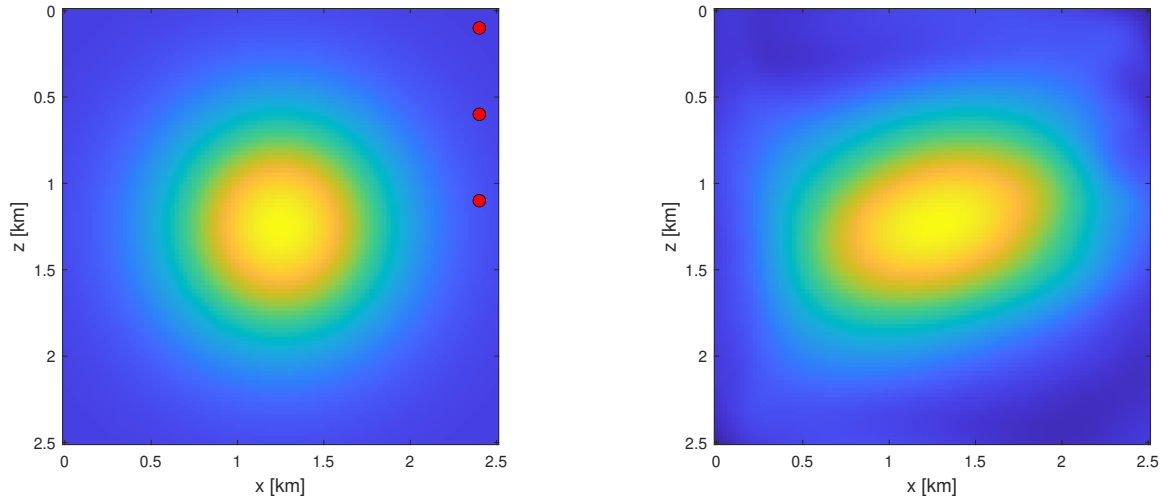
We choose the set of sensor positions, $\mathcal{P}$, shown in Figure 4.6.8 (a), which produces the FWI reconstruction ($\boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P})$) in Figure 4.6.8 (b). We note that the set of sensor positions here is not chosen to produce a good quality reconstruction but rather to illustrate the result of the theorem more clearly.



(a) $\mathcal{P}$ (b) $\boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P})$

Figure 4.6.8: *Set of sensor positions* $\mathcal{P}$ *and correspsonding reconstruction* $\boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P})$

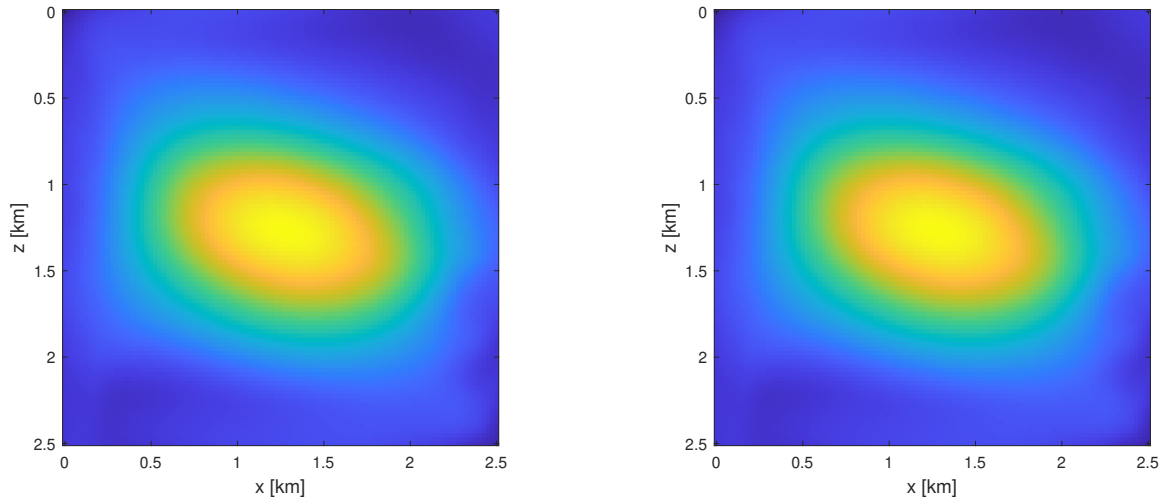
We reflect the sensor positions about the line of reflection shown in Figure 4.6.7 to get the reflected set of sensor positions, $\widetilde{\mathcal{P}}$, shown in Figure 4.6.9 (a). The corresponding reconstruction, $(\boldsymbol{m}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}}))$, is shown in Figure 4.6.9 (b).



(a) $\widetilde{\mathcal{P}}$ (b) $\boldsymbol{m}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}})$

Figure 4.6.9: *Set of sensor positions* $\widetilde{\mathcal{P}}$ *and correspsonding reconstruction* $\boldsymbol{m}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}})$

To demonstrate the result of the theorem, we reflect the reconstruction in Figure 4.6.9 (b) about the line of reflection to get $\widetilde{\boldsymbol{m}}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}})$. We include a side by side comparison of $\widetilde{\boldsymbol{m}}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}})$ with $\boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P})$ in Figure 4.6.10 which demonstrates that the reconstructions appear identical. We find that the infinity norm of the difference between the models is 1.9688×10^{-12}, showing that Theorem 4.3.6 holds in practice, up to some numerical error.



(a) $\boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P})$ (b) $\widetilde{\boldsymbol{m}}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}})$

Figure 4.6.10: *Comparison of reconstructions showing that* $\boldsymbol{m}^{FWI}(\mathcal{S}, \mathcal{P}) = \widetilde{\boldsymbol{m}}^{FWI}(\mathcal{S}, \widetilde{\mathcal{P}})$.

Experiment 3: Demonstrating Corollary 4.4.5

We demonstrate this corollary with a simple sensor optimisation example involving one training model, three sources and three sensors. We ensure that the following assumptions for Corollary 4.4.5 are true in this experiment:

- The domain is symmetric (Assumption 4.2.1 Point 1).
- There is a symmetric layout of sources (Assumption 4.4.2 Point 1).
- The training model is symmetric (Assumption 4.4.2 Point 2).

The positions of the sources are overlayed on the training model in Figure 4.6.11. The line of symmetry is at z =1.25 km. The training model has a symmetric region of higher wavespeed in the centre, with a maximum of 2100 ms^{-1}, that smoothly decreases outward. The domain is of size 2.5 km $\times$ 2.5 km, and is discretised into a 101 $\times$ 101 grid, resulting in 10201 model parameters. We note that this is the same training model and source layout used in Section 3.6, in which we optimised the sensor positions and Tikhonov regularisation parameter. Therefore, we choose the Tikhonov regularisation parameter a priori to be constant at its optimal value ($\alpha = 3.6441$) throughout the sensor optimisation here. In Remark 4.3.2 we made a note about not including the Tikhonov regularisation term in the analysis of this chapter. However, we include the Tikhonov term in this experiment, since the line of symmetry is the horizontal centreline and we are using a uniform discretisation grid, so all the symmetry results still hold.

Figure 4.6.12 displays the starting sensor positions, which are chosen to be symmetric, and the optimal sensor positions found by our bilevel algorithm, which are also symmetric. Although the upper-level solution found may not be unique, this numerical result tells us that at least one of the solutions of the sensor optimisation problem is symmetric.

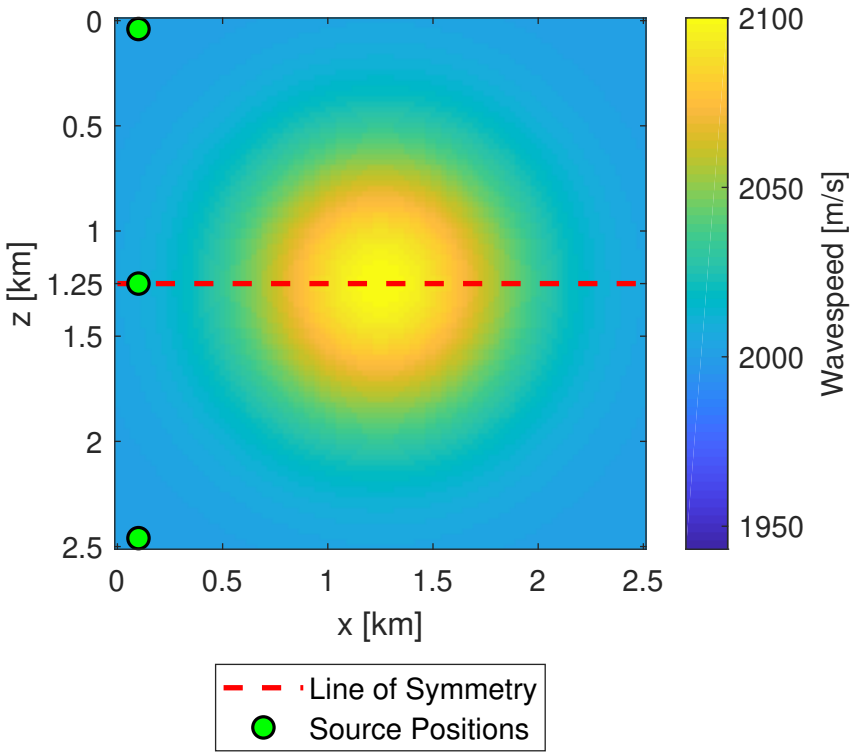


Figure 4.6.11: *Symmetric training model* $\boldsymbol{m}'$ *and symmetric layout of sources*

The FWI reconstruction at the starting guess and FWI reconstruction at the optimised sensor positions are shown in Figure 4.6.13.

Figure 4.6.14 shows the coordinates of the sensors at each iteration as they are being optimised. Subfigure (a) displays the z-coordinates of the sensors. This subfigure shows that the middle sensor (sensor 2), whose starting guess is along the line of symmetry, remains along the line of symmetry at every iteration. The z-coordinates of the other two sensors (sensors 1 and 3) are reflections of each other at every iteration. Subfigure (b) displays the x-coordinates of the sensors. At every iteration, sensors 1 and 3 are at the same x-coordinate, while the x-coordinate of sensor 2 is independent of the other two sensors.

The numerical results obtained by applying the bilevel learning algorithm show that when we have a symmetric domain, symmetric source positions and symmetric training model, then the optimal sensor positions are symmetric. Therefore, the result of Corollary 4.4.5 holds in practice.

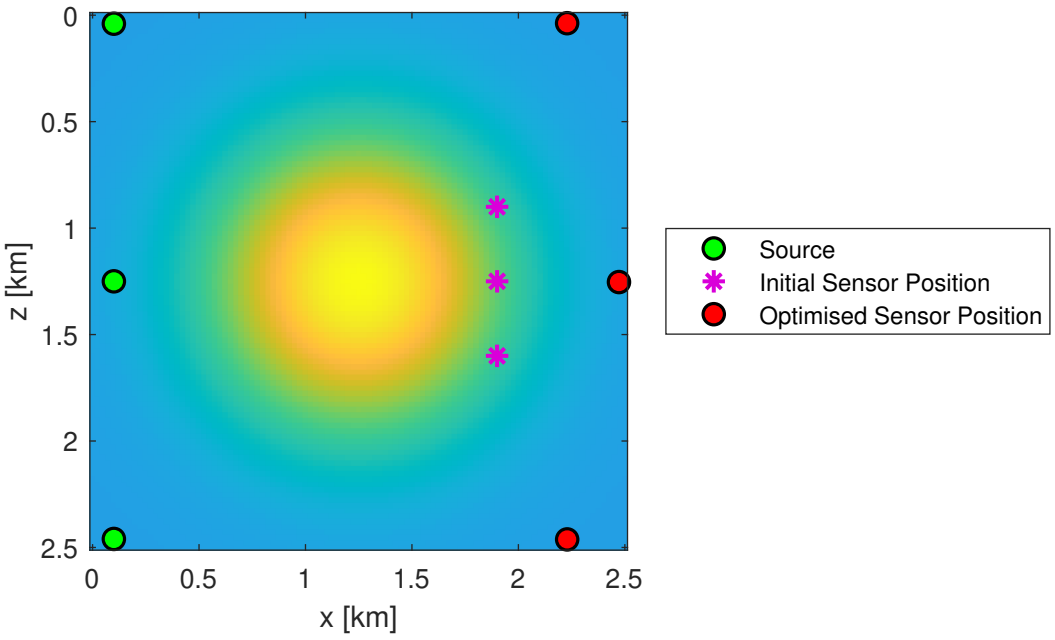
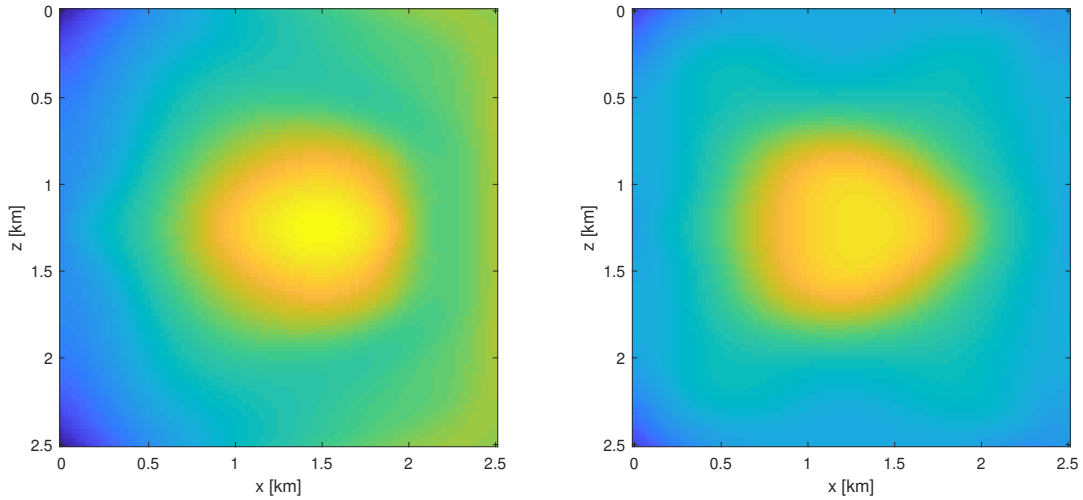
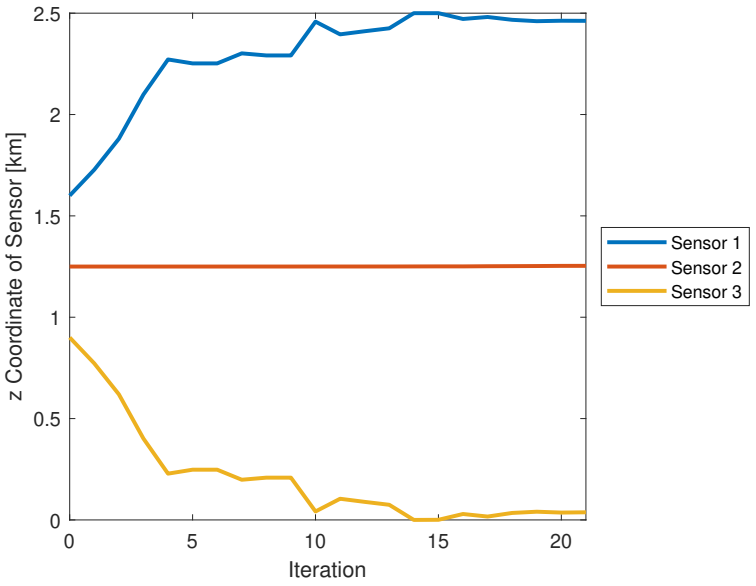


Figure 4.6.12: *Initial and optimised sensor positions.*

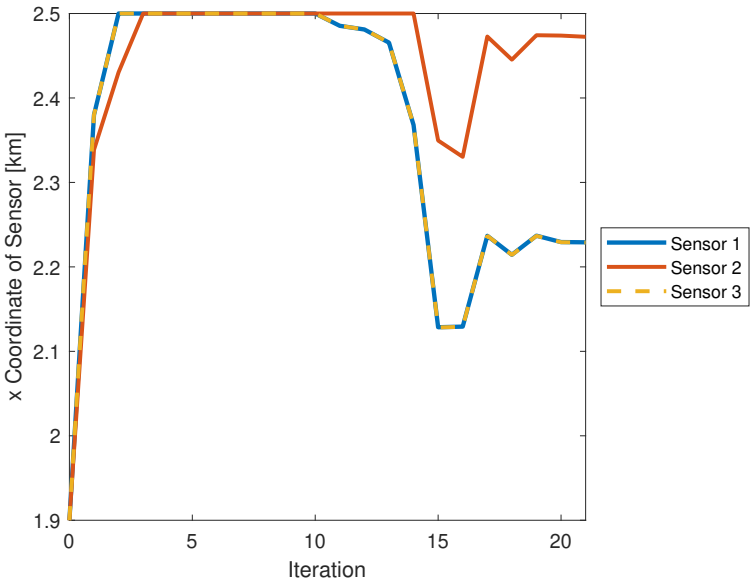


(a) $\boldsymbol{m}^{FWI}(\mathcal{P}_0)$ (b) $\boldsymbol{m}^{FWI}(\mathcal{P}_{\min})$

Figure 4.6.13: *FWI reconstructions at the initial guess,* $\boldsymbol{m}^{FWI}(\mathcal{P}_0)$*, and optimised sensor positions* $\boldsymbol{m}^{FWI}(\mathcal{P}_{\min})$*.*



(a) z-coordinates



(b) x-coordinates

Figure 4.6.14: *Sensor positions versus iteration.*

Experiment 4: Exploiting Symmetry of the Sensors in the Sensor Optimisation Algorithm

We demonstrate how the result of Corollary 4.4.5 can be used to make the sensor optimisation algorithm less computationally expensive. We have seen in the previous experiment that if we have a symmetric domain, training model and layout of sources, then a solution to the sensor optimisation problem is symmetric. There may be other

solutions as we have no guarantee that the sensor optimisation problem is convex here, however, if the initial guess at sensor positions is symmetric, we expect that the solution converged to will also be symmetric (as there is no asymmetry introduced to the problem). We can use this fact to reduce the number of parameters being optimised.

We illustrate the potential reduction in the number of optimisation parameters for the case of the previous problem in Experiment 3. Figure 4.6.15 shows an example of a symmetric set of sensors. The x-coordinates of sensors 1 and 3 are equal, and so instead of optimising the x-coordinates of both of these sensors, we only need to optimise the x-coordinates of one. The distance in the z direction between sensor 1 and the line of symmetry and between sensor 2 and the line of symmetry is equal. Therefore, instead of optimising the z-coordinates of these two sensors, we can instead optimise the distance Δ. The z-coordinate of sensor 2 is lying on the line of symmetry, and therefore will not move from here in order to preserve symmetry. Hence, for this example, instead of optimising 6 parameters (the x- and z-coordinates of 3 sensors), only 3 parameters need to be optimised. This reduced parameter approach generalises to any number N_r of sensors, where the number of optimisation parameters in two-dimensions is reduced from $2N_r$ to N_r. The reduction in optimisation parameters reduces the overall number of PDEs that need to be solved during the sensor optimisation algorithm. As we can see in (3.4.18), the overall number of PDE solves depends on the number of optimisation parameters, so that decreasing the number of optimisation parameters decreases the number of PDEs to be solved. However, the overall number of PDE solves is not directly proportional to the number of optimisation parameters, so halving the number of optimisation parameters does not halve the number of of PDE solves, as we see in the examples.

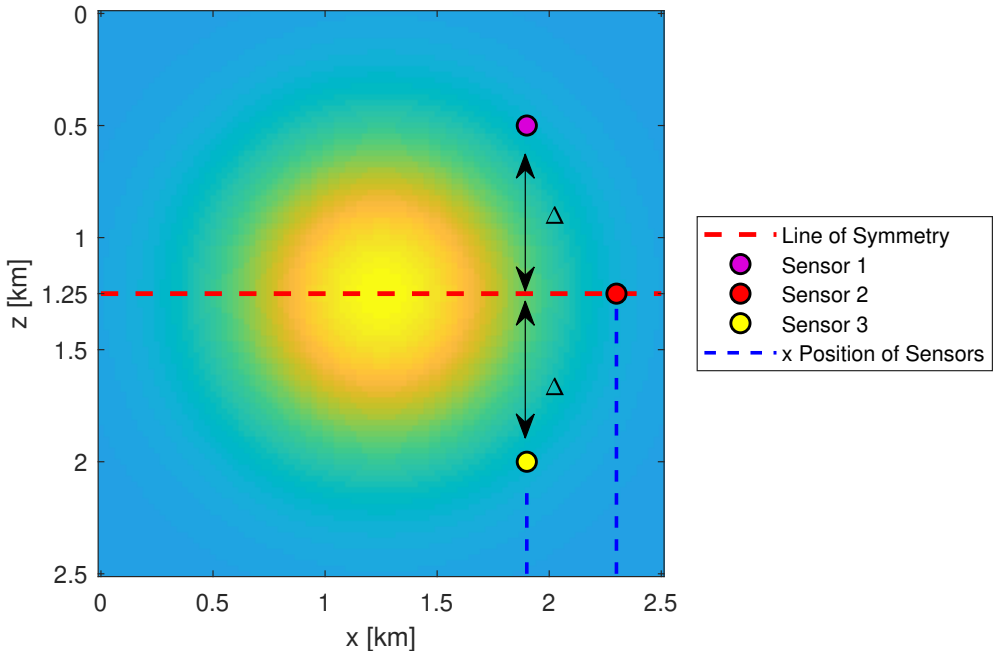


Figure 4.6.15: *Diagram of symmetric example showing how the number of optimisation parameters may be reduced.*

In the following two examples, we illustrate how this reduced parameter approach can be used to arrive at the same solution as the original approach, but faster.

Example 1: We repeat Experiment 3 using the new reduced parameter approach, where only 3 parameters are optimised instead of 6. The optimised sensor positions are overlayed on the training model in Figure 4.6.16. We see that the optimal set of sensor positions matches that found with the original approach in Figure 4.6.12. The optimised reconstruction is therefore the same as that shown in Figure 4.6.13 (b). The new approach was a factor of approximately 1.65 times faster than the original approach.

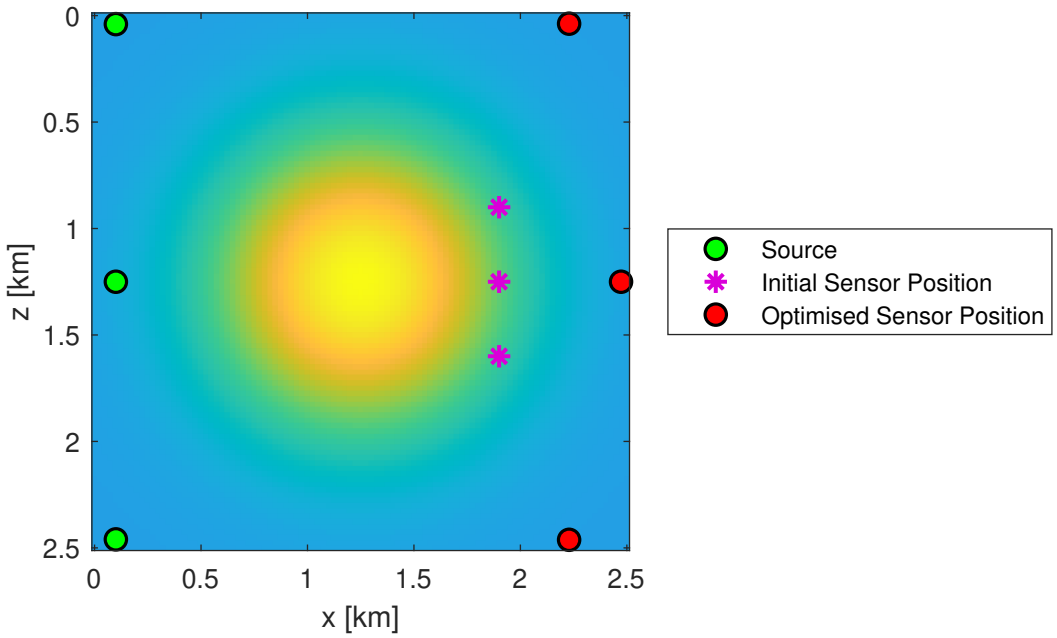


Figure 4.6.16: *Example 1: Initial and optimised sensor positions for the new approach that exploits symmetry.*

Example 2: In this example we solve a sensor optimisation problem with more sensors, using both the original approach of optimising all coordinates and the new approach of optimising half the number of parameters.

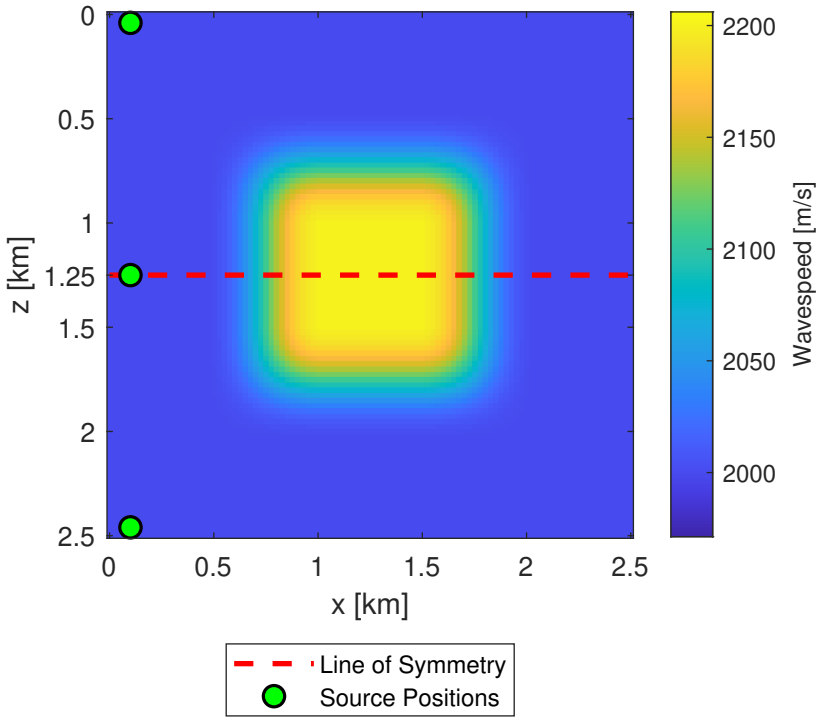


Figure 4.6.17: *Example 2: Symmetric training model* $\boldsymbol{m}'$ *and symmetric layout of sources.*

The symmetric training model and symmetric layout of sources is shown in Figure 4.6.17. The line of symmetry is at $z = 1.25$ km. The training model involves a smooth square area of higher wavespeed (2200 ms^{-1}), surrounded by an area of lower wavespeed (2000 ms^{-1}). Like in Example 1, the domain is of size 2.5 km $\times$ 2.5 km and is discretised into a 101 $\times$ 101 grid. The Tikhonov regularisation parameter is chosen constant at $\alpha = 1.25$. We aim to optimise the positions of 7 sensors. In the original approach, this involves optimising 14 coordinates. The initial guess at sensor positions and optimised sensor positions are shown in Figure 4.6.18, and the corresponding initial and optimised reconstructions are shown in Figure 4.6.19. In the new reduced parameter approach, we only need to optimise 7 coordinates. We see in Figures 4.6.20 and 4.6.21 that we achieve the same optimal sensor setup and the same reconstructions as the original approach. However this result is achieved a factor of approximately 1.38 times faster.

These experiments show that the symmetry properties of a problem can be exploited to find an optimal set of sensor positions in a computationally cheaper manner.

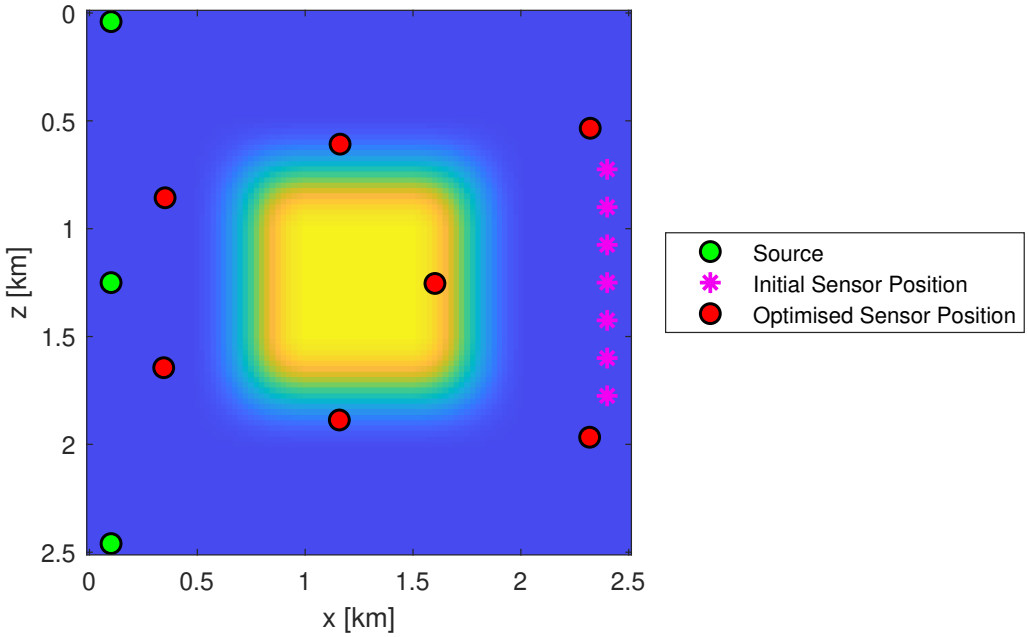
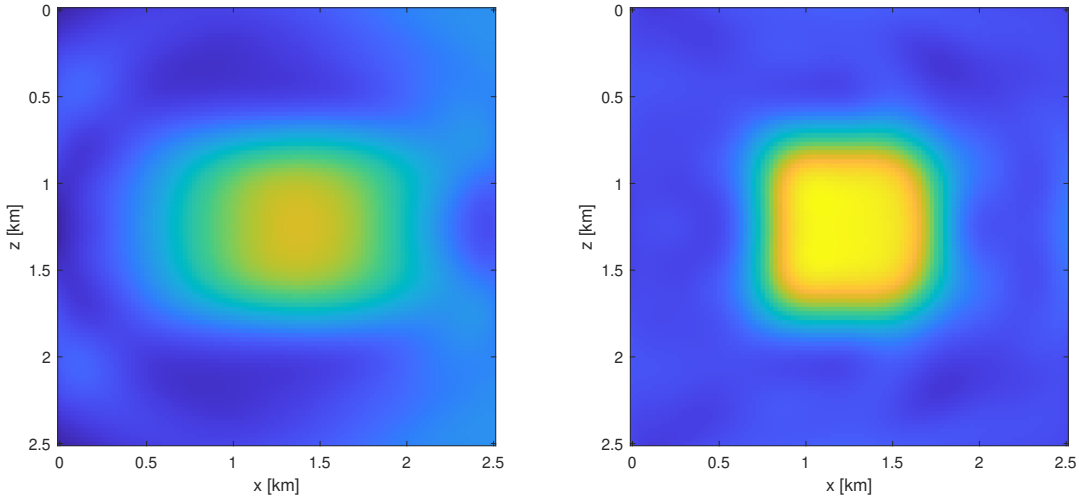


Figure 4.6.18: *Example 2: Initial and optimised sensor positions for the original sensor optimisation approach.*



(a) $\boldsymbol{m}^{FWI}(\mathcal{P}_0)$ (b) $\boldsymbol{m}^{FWI}(\mathcal{P}_{\min})$

Figure 4.6.19: *Example 2: FWI reconstruction at the initial guess,* $\boldsymbol{m}^{FWI}(\mathcal{P}_0)$*, and optimised sensor positions* $\boldsymbol{m}^{FWI}(\mathcal{P}_{\min})$ *for the original sensor optimisation approach.*

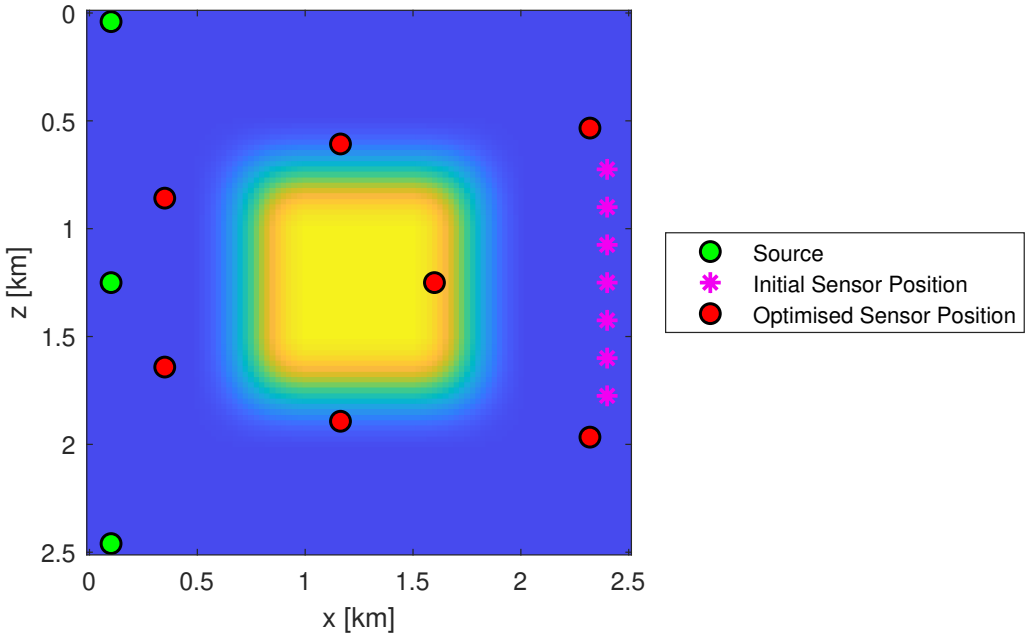
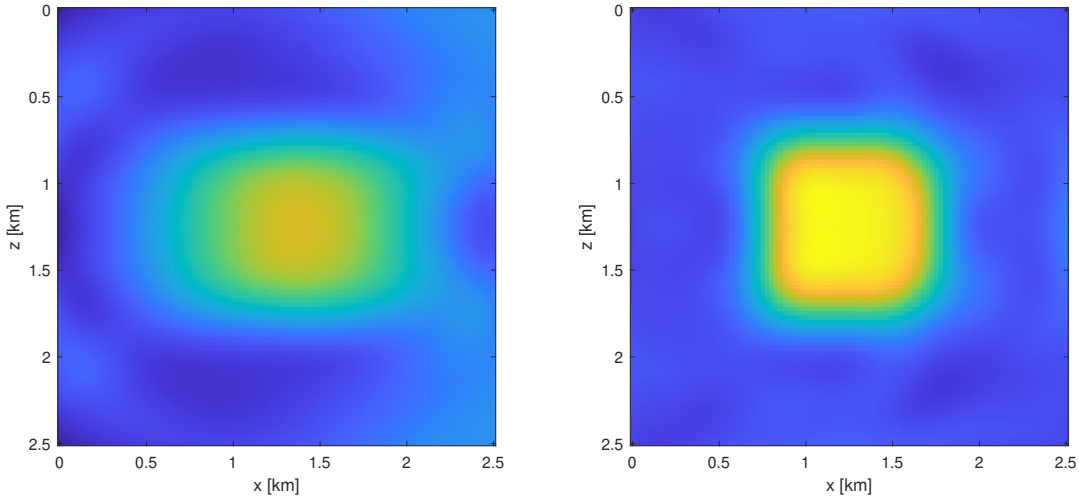


Figure 4.6.20: *Example 2: Initial and optimised sensor positions for the reduced parameter sensor optimisation approach.*



(a) $\boldsymbol{m}^{FWI}(\mathcal{P}_0)$ (b) $\boldsymbol{m}^{FWI}(\mathcal{P}_{\min})$

Figure 4.6.21: *Example 2: FWI reconstruction at the initial guess,* $\boldsymbol{m}^{FWI}(\mathcal{P}_0)$*, and optimised sensor positions* $\boldsymbol{m}^{FWI}(\mathcal{P}_{\min})$ *for the reduced parameter sensor optimisation approach.*

4.7 Application of Symmetry Results

All experiments in this section were performed on synthetic models that were designed to be symmetric. In real world geophysical problems, subsurfaces will generally not be naturally symmetric. However, there are other FWI applications where the symmetry results presented here can be used for the optimal sensor placement problem. One application would be FWI imaging in a medical setting where symmetry is expected in the image, exploiting the high level of symmetry present in the human body. As an example, the brain is largely symmetrical (not perfectly symmetrical however) [118], and FWI has been applied to image the brain in [81]. Our symmetry results could therefore be useful in optimising sensor placement for the FWI imaging of the brain. Another potential application of our symmetry results would be in the field of structural health monitoring. FWI has been used in ultrasonic non-destructive testing, for example see [159], and so our theory could be used in the optimisation of sensor placement for the ultrasonic non-destructive testing of symmetric structures, for example in the monitoring of pipe corrosion or in the inspection of railway tracks.

Chapter 5

Algorithms and Implementation

Chapter Summary: This chapter presents the algorithms that we have designed to solve the bilevel problem, as well as implementation details of these algorithms. The implementation details include novel techniques developed to improve the efficiency of the algorithms. In §5.1, we discuss avoiding local minima on both the upper- and lower-levels through the use of a new bilevel frequency continuation algorithm.

In §5.2, we discuss an aspect of the upper-level gradient computation - in particular how to solve the linear system involving the FWI Hessian arising in the gradient formula. We investigate how the number of iterations taken to solve this system varies with different parameters and show that the number of iterations is, in general, large, and increases as the discretisation grid size is refined. We have established two novel preconditioning strategies (§5.2.2) to reduce the number of iterations taken to solve this system. We show that both preconditioning strategies work effectively to speed up the solution of the linear system, producing a reduction in the number of iterations by up to 96%, and when used within the bilevel algorithm, reducing the computational time by several hours in some cases. We also investigate of how each preconditioning strategy is affected by various parameters, hence demonstrating that the number of iterations taken to solve the preconditioned system is unaffected by grid size. We include a breakdown of the cost of each preconditioning strategy and make recommendations about which strategy to use.

In §5.3 we discuss parallelising the bilevel algorithm and show through measurements of computation time that the algorithm scales well in parallel. In (§5.3.3) we provide an overview of the computational time spent on different parts of the algorithm and a detailed description explaining what we have observed.

The full bilevel sensor placement optimisation algorithm is included in §5.4. The implementation of the restriction operator and its importance in ensuring the smoothness of the upper-level objective function is also discussed here. The bilevel frequency continuation strategy, preconditioning strategy, the implementation details of the restriction operator and the algorithms presented in this chapter (excluding the FWI and preconditioned conjugate gradient algorithms) are all novel. We note here that, since this section is mainly relevant to implementation, the variables will be written in their discrete format.

5.1 Bilevel Frequency Continuation

Context and goal: In Section 2.5.1, we described the process of frequency continuation in FWI (the lower-level problem in the bilevel problem Definition 3.3.4), which is standard practice for avoiding local minima in frequency domain FWI. We recall from this discussion that the FWI objective function is smoother for lower frequencies, but that a range of frequencies are required to reconstruct a range of different sized features in the image. Hence higher frequencies are used to help improve the quality of the FWI image. Here we present a novel bilevel frequency continuation approach that involves continuation on both the upper and lower-level. The goal of our bilevel frequency continuation algorithm is to, as far as possible, avoid local minima on both levels of the bilevel problem, hence making it easier for a local optimisation method to find the global optimal solution.

Motivating Example: We motivate our approach with the following example. Figure 5.1.1 is an example of a training model, with three sources and three sensors used for acquisition. We consider the sensors to be constrained along a line, i.e., they can move in one-dimension only. Due to the symmetry of the setup, and the theory present in Chapter 4, it is reasonable to place the sensors symmetrically about the centre horizontal line and to consider only one optimisation variable – the distance Δ between the top and bottom sensor from the line of symmetry. The parameter Δ, i.e., sensor distance from the centre, ranges from 0, meaning all sensors overlapping at the centre, to L, meaning the top and bottom sensors are at the edges of the domain. We vary Δ from 0 to L (in 1250 steps), perform FWI using synthetic data for that setup at each step, and compute the upper-level objective function ψ (3.3.1) for that point. This process is completed for a lower frequency (0.5 Hz) and a higher frequency (7 Hz), giving us two plots of the upper-level objective function for this problem, shown in Figure 5.1.2. The upper-level objective function for the lower frequency problem is smooth and has one minimum. An objective function of this form is easily handled by a local optimisation method. The upper-level objective function for the higher frequency problem has multiple local minima, but has a global minimum that is not far from the global minimum of the lower frequency problem. For the high frequency objective function, unless the starting guess of the local optimisation method is close to the global minimum, the global minimum would not be found by a local optimisation method. This example demonstrates that ψ is smoother for lower frequencies, and illustrates the need for a good starting guess for higher frequency sensor optimisation problems. This motivates our bilevel frequency continuation approach.

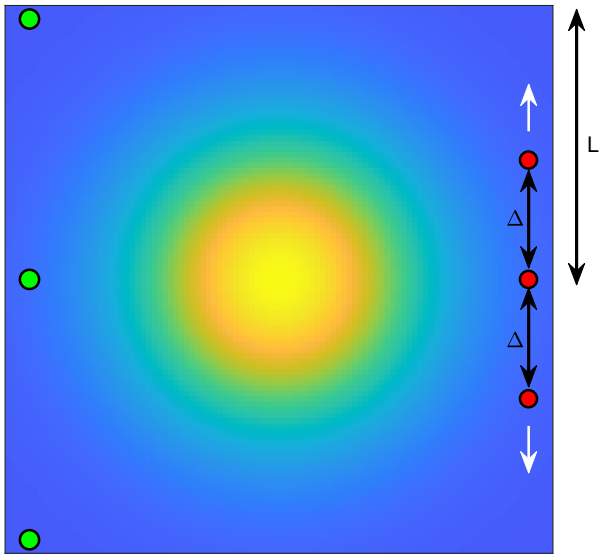


Figure 5.1.1: *Setup used in plots of upper-level objective function. The symbol* ● *represents a source, and the symbol* ● *represents a sensor. The outer sensors are moved along a line to produce the plots in Figure 5.1.2.*

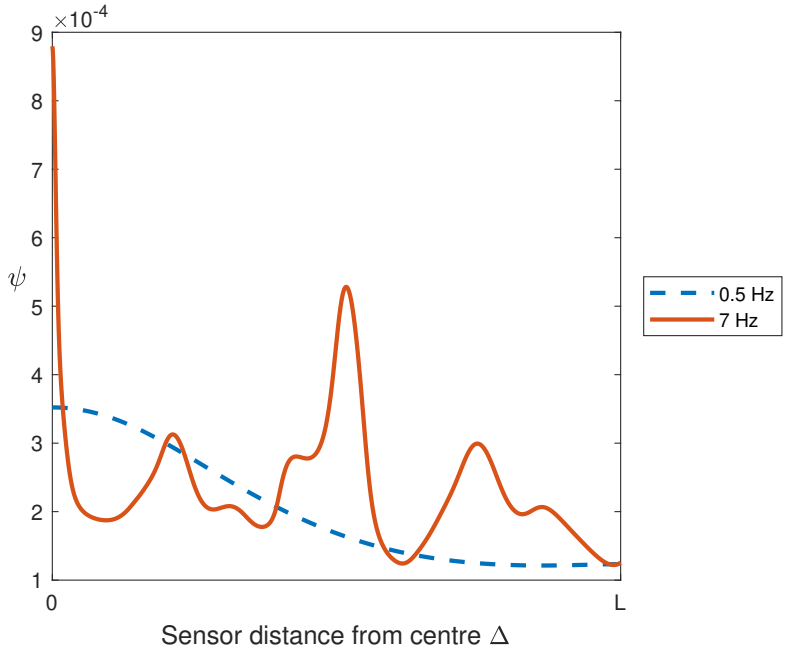


Figure 5.1.2: *Upper-level objective function ψ for a low and higher frequency problem.*

We illustrate the steps of our proposed bilevel frequency continuation algorithm in Figure 5.1.3. Each row of Figure 5.1.3 shows a plot of the upper-level objective function ψ for the problem setup in Figure 5.1.1, starting at a low frequency on row one, and increasing to progressively higher frequencies/frequency groups on rows two and three. In Subfigure (a) we represent a typical starting guess for the parameter to be optimised, Δ, by an open red circle. As this is a low frequency problem, ψ is smooth and has one minimum, and therefore the sensor placement optimisation problem can be solved straightforwardly. The solution is shown in (b) by the closed red circle. This solution gives us a rough estimate of the overall optimal sensor position. We then progress to a group of higher frequencies, and use the solution to the low frequency problem as a starting guess (shown in (c)). This problem can be solved straightforwardly due to the good starting guess. The solution (shown in (d)) improves the original estimate of the optimal sensor position. This process is continued, progressing through higher frequencies and iteratively improving the estimate of optimal sensor position each time. Subfigures (e) and (f) demonstrate the power of the bilevel continuation approach to avoid the multiple local minima and find the global minimum (which in this case is the sensors being spread out to near the edges of the domain).

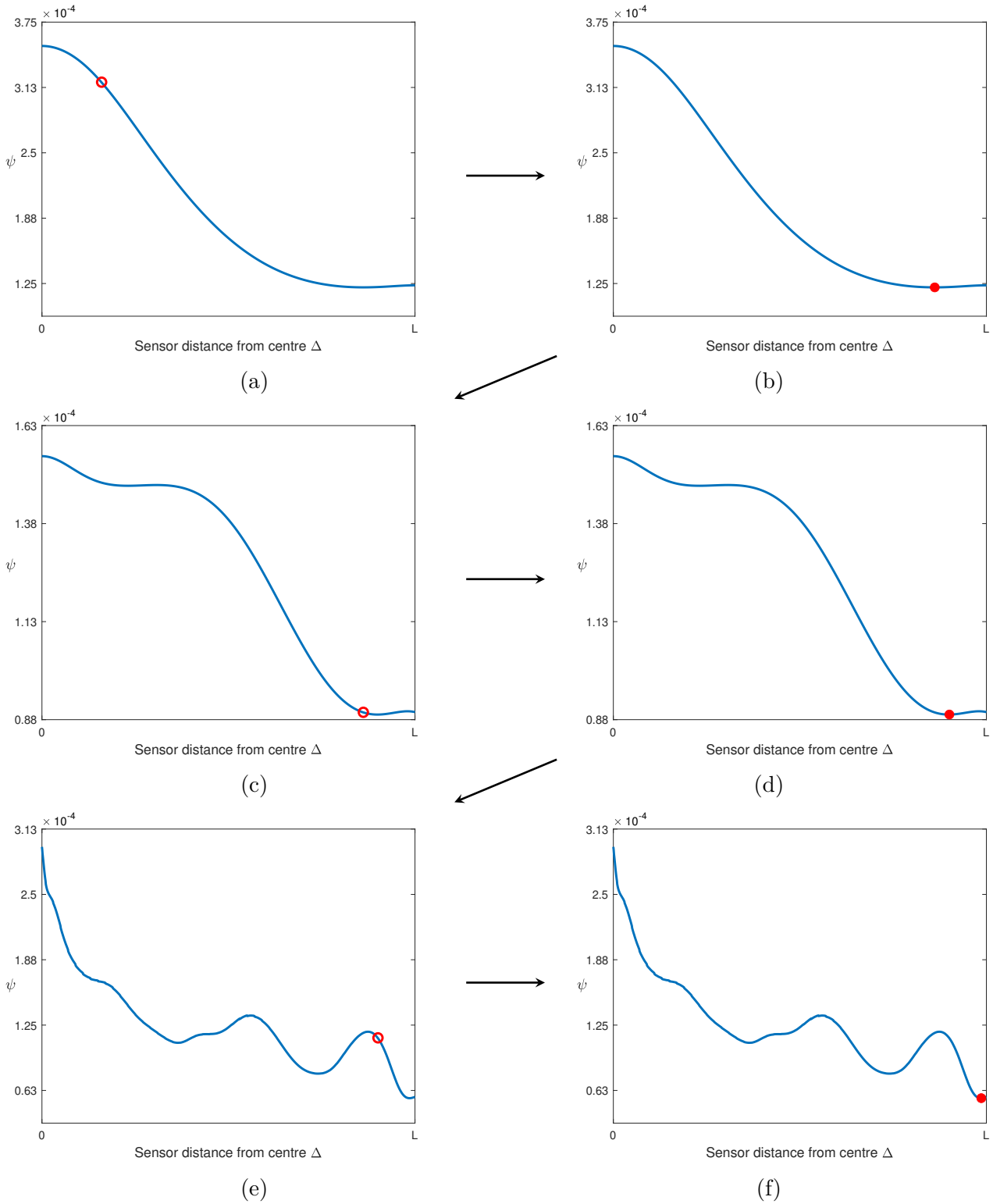


Figure 5.1.3: *Plots of the upper-level objective function in an illustration of the bilevel frequency continuation approach. The symbol* ○ *denotes a starting guess, and the symbol* ● *denotes a minium.*

The Algorithm: We present these steps formally in Algorithm 5.1.1. The algorithm involves grouping frequencies into groups of increasing order, looping over the frequency groups and solving the full bilevel problem (presented later in Algorithm 5.4.2) on each loop. The solution to the upper and lower-level problems (i.e., the optimal sensor positions and optimal models) for the first frequency group are used as the starting guesses for the next, higher, frequency group. In this way, we can see how the frequency continuation works simultaneously for both the upper and lower-levels. We write Algorithm 5.1.1 for one training model only for simplicity here. In Section 5.4 we write this algorithm more generally for a training set of any size.

Algorithm 5.1.1 Bilevel Frequency Continuation

1: *Inputs:* $\boldsymbol{p}_0$, $\boldsymbol{m}_0$, $\{\omega_1 < \omega_2 < ... < \omega_{N\omega}\} \in \mathcal{W}$, $\boldsymbol{m}'$
2: Group frequencies into N_f groups $\{g_1, g_2, \ldots, g_{N_f}\}$
3: **for** $k = 1$ **to** N_f **do**
4: $\quad [\boldsymbol{p}_{\min}, \boldsymbol{m}^{FWI}] \leftarrow$ Bilevel Optimisation Algorithm
5: $\quad \boldsymbol{p}_0 \leftarrow \boldsymbol{p}_{\min}$
6: $\quad \boldsymbol{m}_0 \leftarrow \boldsymbol{m}^{FWI}$
7: **end for**
8: *Output:* $\boldsymbol{p}_{\min}$

Remark 5.1.1. *We note that it is possible to incorporate a multilevel minimization approach, such as that detailed in [130, Section 9.4], into the bilevel frequency continuation approach. This approach would involve solving the lower frequency problems with a coarser discretisation and progressively refining the discretisation as we progress to increasingly higher frequency groups. This method would potentially reduce memory requirements and computing time. We note that it is important that care is taken so that discretisation is not too coarse due to the nature of numerical error in wave propagation problems. We don't implement this multilevel minimization approach in this thesis since, to ensure our forward modelling step (performed with a low-order finite difference scheme) is accurate, we avoid mesh coarsening.*

Experiment 1: We apply the bilevel frequency continuation algorithm to the one-parameter sensor placement optimisation problem that we have been focusing on (Figure 5.1.1). The symmetric training model and the line along which the sensors are constrained are shown in Figure 5.1.4. The model is discretised into a 101×101 grid.

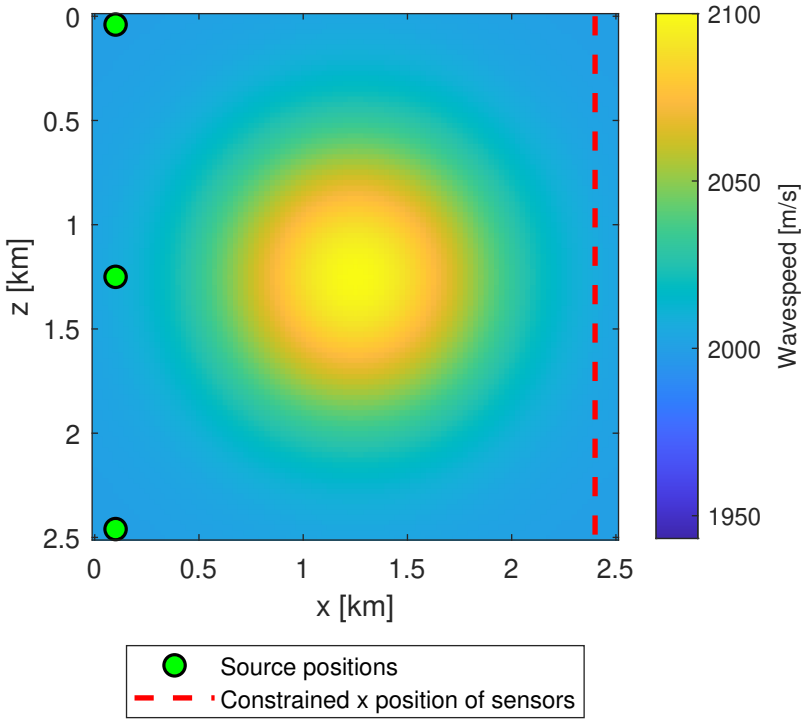


Figure 5.1.4: *Training model and setup for Experiment 1.*

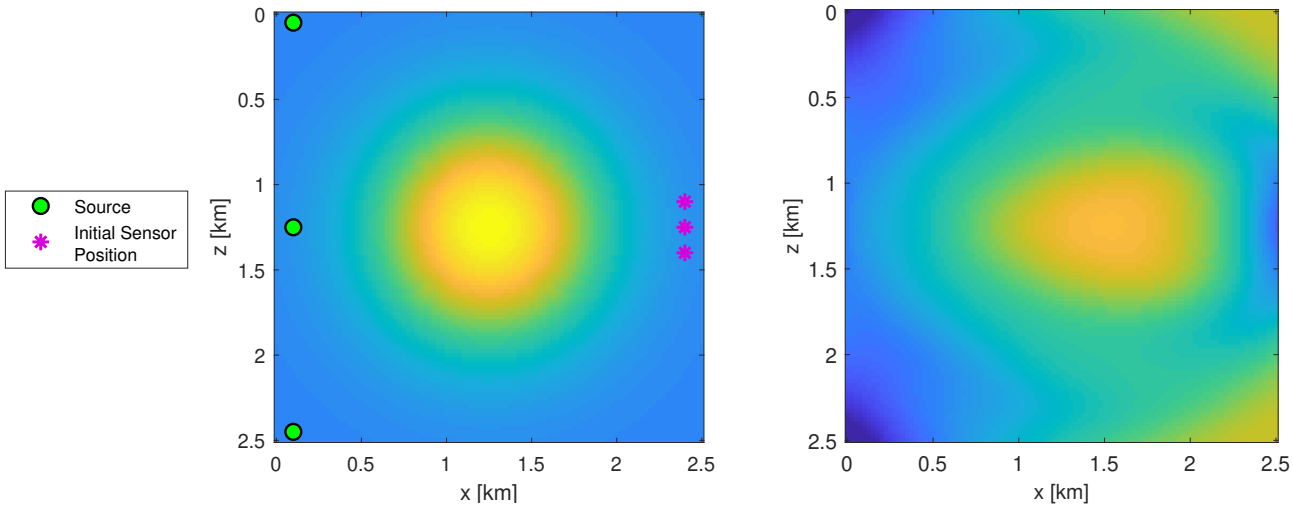
Figure 5.1.5 shows the initial guess for sensor positions and the FWI reconstruction produced by this initial sensor setup. The corresponding value of ψ is 9.1429×10^{-5}.

The result of applying the bilevel frequency continuation algorithm is shown in Figure 5.1.6. As expected, the optimal sensor positions are spread out such that the top and bottom sensor are near the edges of the domain (to be more specific, they are approximately 26 metres away from the edge of the domain). The optimal value of ψ here is 1.3246×10^{-5}, and therefore the improvement factor (defined in (3.6.1)) is 6.9. (We note that the improvement factor is small relative to some other examples in this thesis due to the fact that the x positions of the sensors are fixed here).

The result of the sensor placement optimisation algorithm *without* using the bilevel frequency continuation approach is shown in Figure 5.1.7. The sensors get stuck in a local minimum (on the centre line/line of symmetry). The resulting value of ψ is 6.4329×10^{-5}, giving an improvement factor of only 1.4213.

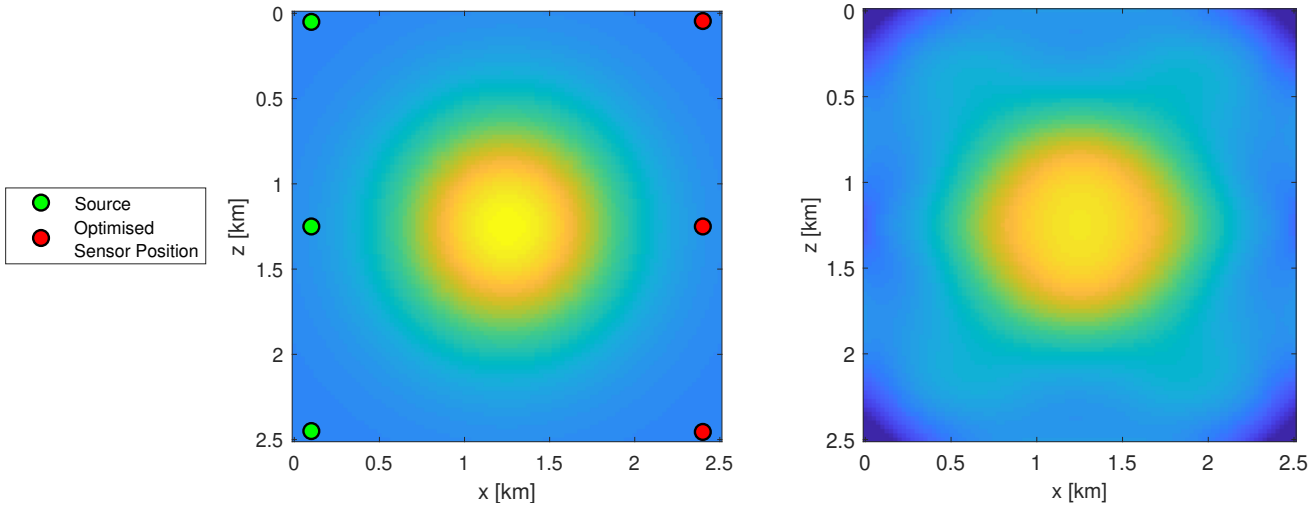
In this experiment, we have demonstrated that the continuation algorithm has significantly improved the results of the bilevel learning algorithm by finding what appears to be the global minimum for this problem. Applying the bilevel algorithm

without using our continuation technique results in getting stuck in a local minimum. *In conclusion, this example shows advantage of using the bilevel frequency continuation algorithm.*



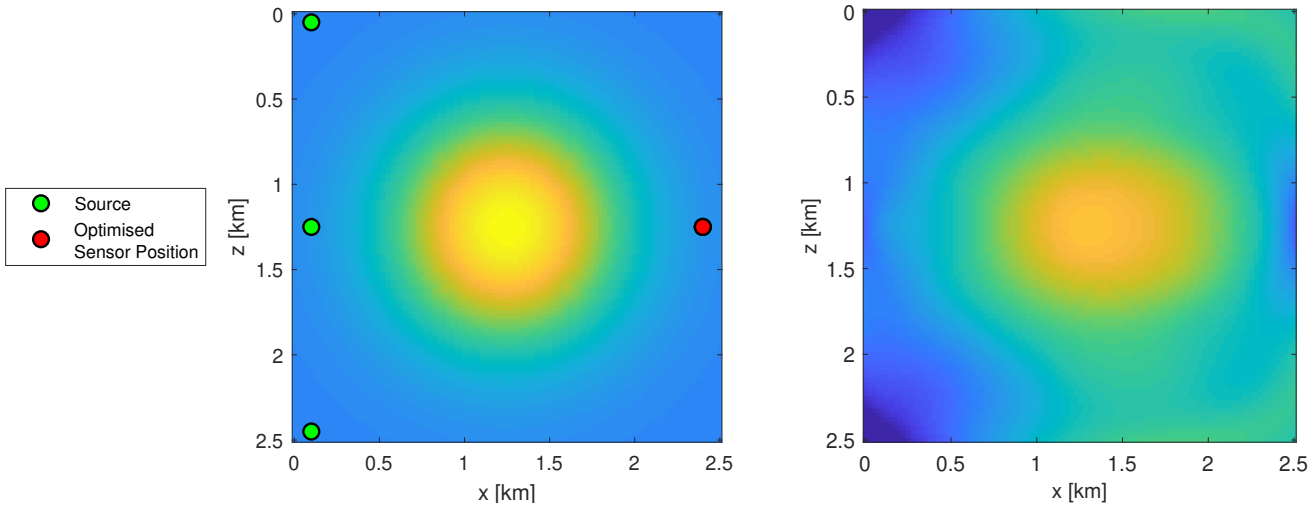
(a) Initial Sensor Positions $\boldsymbol{p}_0$ (b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0)$

Figure 5.1.5: *Initial guess for sensor positions and the resulting reconstruction.*



(a) Optimised Sensor Positions $\boldsymbol{p}_{\min}$ (b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min})$

Figure 5.1.6: *Optimised sensor positions and the resulting reconstruction using the bilevel frequency continuation algorithm.*



(a) Optimised Sensor Positions $\boldsymbol{p}_{\min}$ (b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min})$

Figure 5.1.7: *Optimised Sensor Positions and the resulting reconstruction without using the bilevel frequency continuation algorithm. The sensors are overlayed in (a) so that three sensors appear as one.*

Choice of Frequencies: The bilevel frequency continuation algorithm used in this example involved three frequencies with an increasing interval between them (0.5, 0.9 and 2.5 Hz). We used these three frequencies to form three frequency groups for bilevel continuation, the first with a single frequency (0.5 Hz) to esnure the smoothest possible objective function, and the following groups with two frequencies each and an overlap between each (i.e., a group (0.5, 0.9)Hz and the final group (0.9, 2.5)Hz). The choice of increasing interval was motivated by the FWI continuation strategy suggested by [166] and the overlap between groups is motivated by the approach for FWI continuation in [77] and [35]. We repeated Experiment 1 with five equally spaced frequencies in the range 0.5 to 2.5 Hz, split into 5 groups with an overlap of one frequency between each group. The bilevel continuation algorithm converges to the exact same optimal sensor setup as the three group case (i.e., the sensors are 26 metres from the edge of the domain), but takes 3.5 times longer for the solution to be found.

This demonstrates that adding more frequency groups can slow the problem down and that it is possible to find the same solution faster with less frequencies, which agrees with what is seen in the standalone FWI problem, as noted by [166].

Experiment 2: This example gives more insight into how the bilevel frequency continuation algorithm works. The training model used here is the same as that used in Experiment 1 (see Figure 5.1.4), but the sensors are free to lie anywhere in the domain. The Tikhonov regularisation parameter is chosen to be $\alpha = 1.25$. In fact, Experiment 2 is the same optimisation problem that is solved in Section 3.6, we just focus more on the details of the frequency continuation here. As in Experiment 1, we choose three frequency groups with an increasing interval between the chosen frequencies. Figure

5.1.8 shows the positions of the sensors at the end of each frequency group. Figures 3.6.2 and 3.6.4 show the initial and the final optimised reconstructions respectively. Clearly the first group (lowest frequency) does most of the work since the sensors move the largest distance in this group (from the 'Initial Sensor Position' to 'Group 1 Solution'). This makes sense as the lowest frequency objective function should be smooth enough that the sensors can move far away from the starting guess. The Group 2 solution and Group 3 solution respectively involve just small corrections of the Group 1 solution. In the second group, the x positions of the outer sensors change noticeably (along with small changes in z) and then in the final group the z positions move outwards. This behaviour is also visualised in Figure 5.1.9. The first frequency group involves most of the movement of the sensors in both the x and z directions, and takes 11 iterations to converge. The second group involves less drastic changes in position and takes 6 iterations to converge, while the final group only takes 2 iterations to converge.

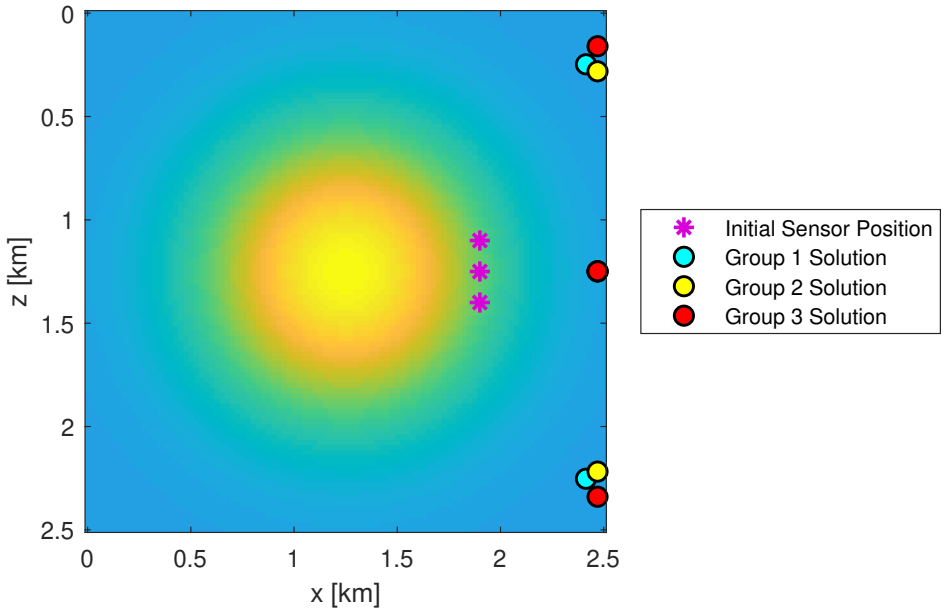
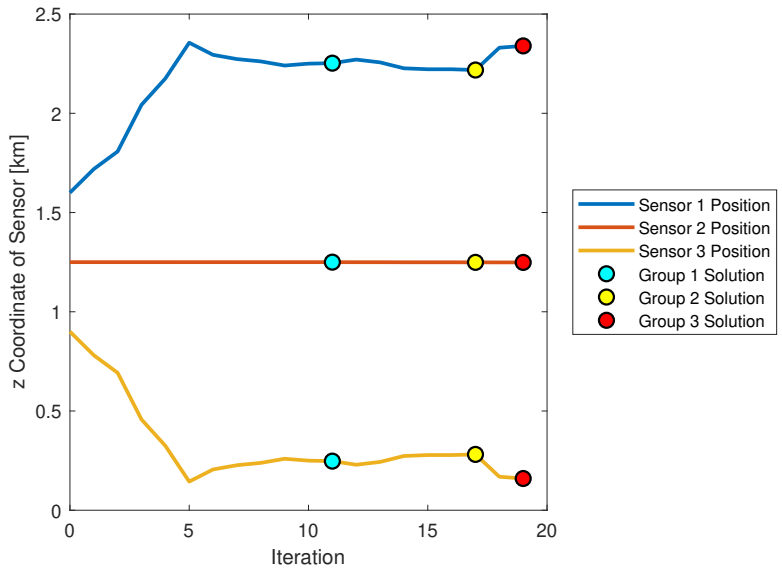


Figure 5.1.8: *Sensor positions converged to in each frequency group for Experiment 2.*

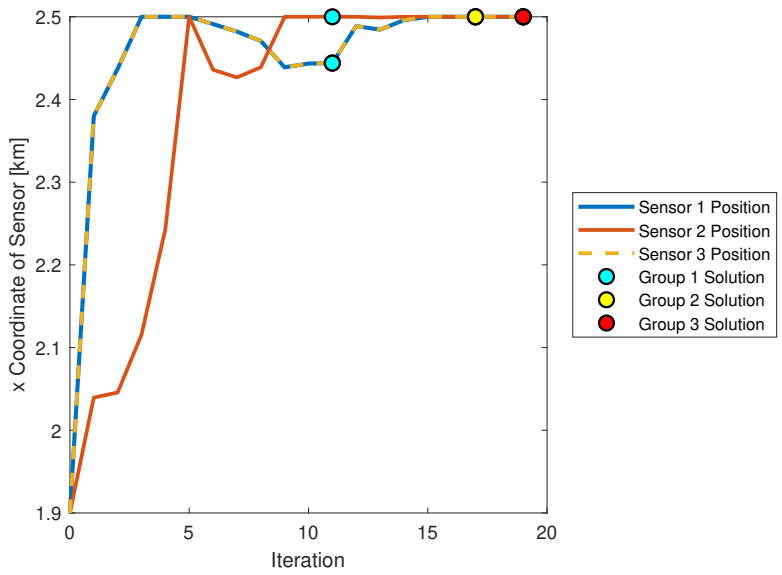
In Figure 5.1.10, we show how the value of the upper-level objective function ψ varies with iteration and across frequency groups. Most of the reduction of ψ occurs in the first group, corresponding to the large change in sensor positions. Progressing to the higher frequency groups results in a reduction in ψ, even without a change in sensor position (i.e., at iterations 11 and 17) due to the lower-level continuation occurring simultaneously with the upper-level continuation. The small corrections to sensor position in those groups correspond to small reductions in ψ.

This example demonstrates the general behaviour that we expect from the bilevel frequency continuation algorithm - most of the work, in terms of movement of sensors and reduction in ψ, is done in the first group, with less and less work required as

we progress through groups, due to the proximity of the starting guess to the optimal solution.



(a) z-coordinates



(b) x-coordinates

Figure 5.1.9: *Sensor position versus iteration for each frequency group.*

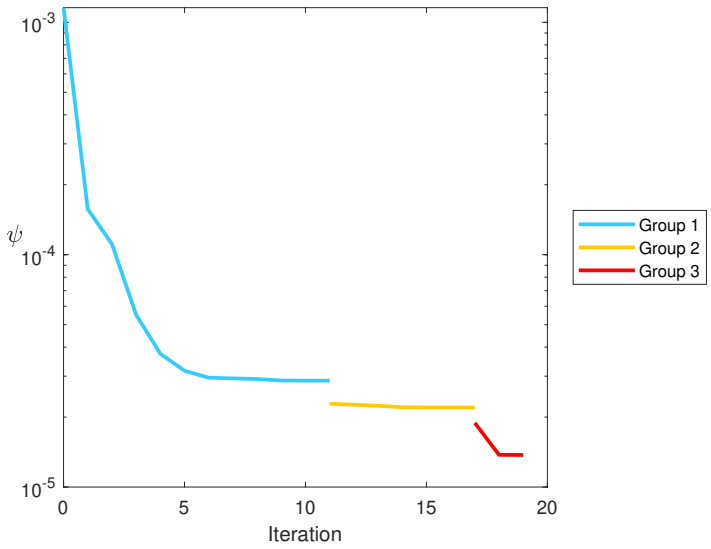


Figure 5.1.10: *Upper-level objective function ψ versus iteration for each frequency group.*

Incorporating the Tikhonov Parameter into Bilevel Frequency Continuation

We now discuss methods for including the optimisation of the FWI Tikhonov parameter α, as discussed in Section 3.5, into our bilevel frequency continuation approach.

To understand how the upper-level objective function ψ varies with α across frequencies, in Figure 5.1.11 we plot ψ for a range of α and a range of frequencies. These plots are for the same problem shown in Figure 5.1.1, where instead of varying the sensor positions, we vary α. We choose a constant symmetric sensor setup, and vary α from 0 to 200, in steps of 0.25. At each step, we perform FWI and compute and plot the value of ψ for that value of α. This is repeated for a range of frequencies. Note that the lowest frequency (0.5 Hz) and highest frequency (7 Hz) in Figure 5.1.11 are the same as that in Figure 5.1.2 (where sensor position is being varied). We see from Figure 5.1.11 that for all frequencies, there is one global minimum for α, which should be able to be found by a local optimisation method from any starting guess. *We can conclude from this experiment that the frequency does not affect the smoothness of ψ versus α.* This is expected as the Tikhonov regularisation term in the FWI objective function does not have any dependency on frequency. Therefore the optimisation of α does not require upper-level frequency continuation in the same way that sensor position does.

However, although we don't require frequency continuation to find the optimal α, we still must optimise this parameter simultaneously with sensor positions to find the best possible sensor position. The reason why is demonstrated in Figure 5.1.12. Here we plot ψ versus sensor position, again for the setup in Figure 5.1.1, for a constant frequency, 4 Hz, and various values of α. Figure 5.1.12 shows that the parameter α affects the shape of ψ, the number and position of local maxima/minima, where the global optimal

sensor positions are, and the value of ψ at those positions. We recall that we saw the effect of this in the experiments in Section 3.6, where the optimal sensor positions were different depending on whether α was being optimised or not. Since α impacts the optimal sensor positions, it should be optimised alongside the sensor positions in some way, and not separately from it.

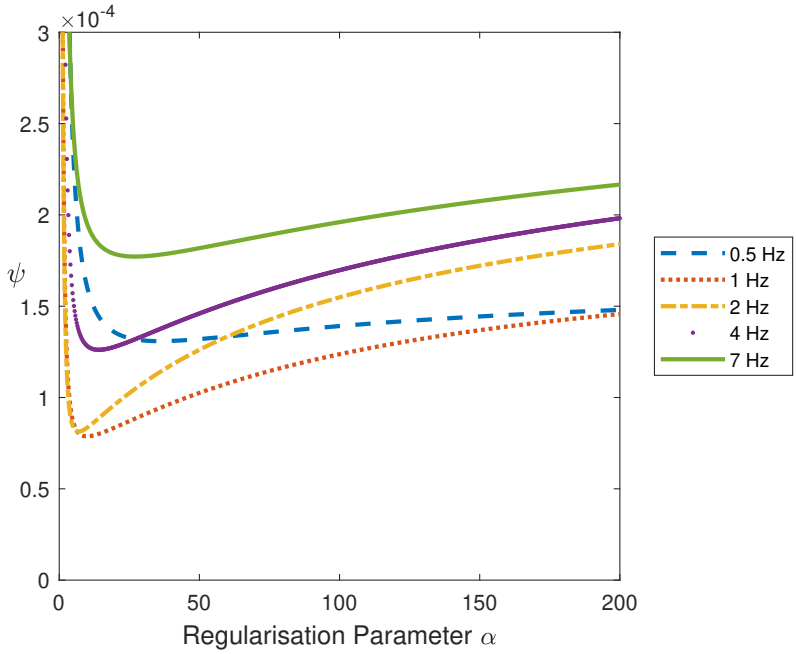


Figure 5.1.11: *Upper-level objective function ψ versus Tikhonov regularisation parameter α for various frequencies.*

We propose three approaches to the problem of incorporating the optimisation of α into the bilevel frequency continuation algorithm. These are:

- **Approach I:** Include α in the full bilevel frequency continuation approach, i.e., optimise α simultaneously with sensor position in every frequency group.
- **Approach II:** Optimise sensors only in the first frequency group, while keeping α constant, and then optimise α simultaneously with sensor position in the following frequency groups. This approach allows the sensors to do most of their large movement in the first group before adding α as an optimisation variable.
- **Approach III:** Optimise sensor positions only while keeping α constant for most of the bilevel frequency continuation approach and include α as an optimisation variable in final frequency group only. This means that there is no continuation in α.

We provide an example of these approaches in the following experiment.

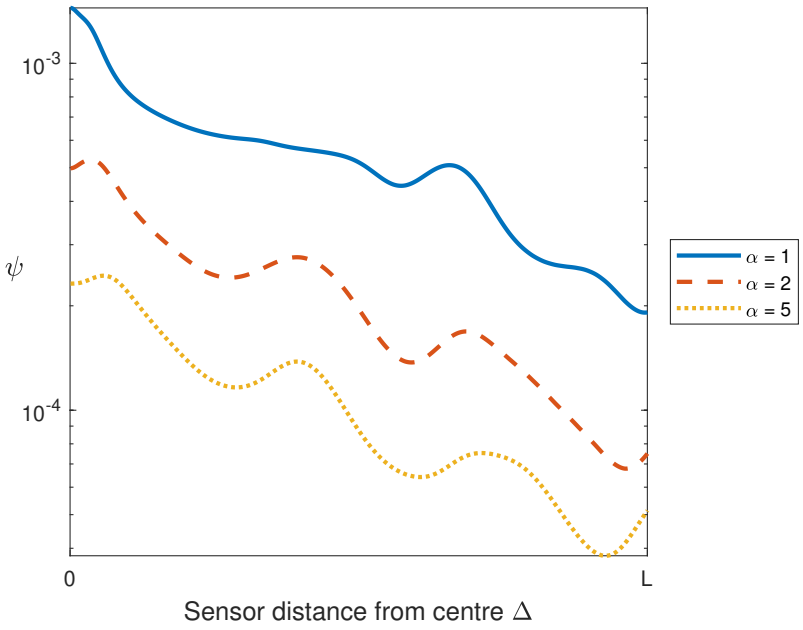


Figure 5.1.12: *Upper-level objective function ψ versus sensor position for various values of α. Frequency is constant at 4 Hz. Note that the y-axis is on a log -scale so that all three plots of ψ are visible clealy.*

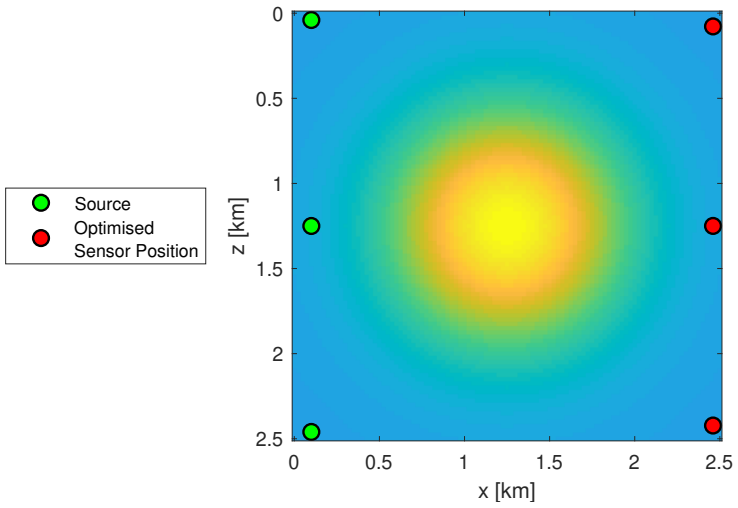
Experiment 3: We repeat Experiment 2 but instead of only optimising sensor positions, we also optimise α. We use each of the three approaches proposed above and compare the results. In Experiment 2 we use three frequency groups, meaning that in this experiment, Approach I involves optimising α in all three groups, Approach II involves optimising α in two groups, and Approach III involves optimising α in the final group only. The starting guess used for α is 1.25.

The result of **Approach I** is shown in Figure 5.1.13. Subfigure (a) displays the optimal positions of the sensors found by Approach I, and Subfigure (b) shows the corresponding FWI reconstruction. Subfigure (c) shows the Tikhonov parameter versus iteration for the whole algorithm, and Subfigure (d) displays the value of the objective function ψ versus iteration, where each group is highlighted. In comparison to Experiment 2 where only sensor positions are optimised, the iteration count has increased significantly. The majority of the iterations still occur in the first group, during which most of the reduction in ψ occurs. We can see from (c) that during the first group the Tikhonov parameter grows very large in the first 19 iterations, reaching a peak of 46.76, before decreasing again and converging to 3.89 at the end of the first group. The parameter α remains relatively close to this value for the rest of the algorithm and eventually converges to 4.88. The large increase of α is the first 19 iterations can be explained as follows. The initial guess for sensors, shown in Figure 5.1.8, is relatively poor, and when the sensor positions are 'bad', the regularisation parameter wants to be higher to compensate for poor data that the sensors are providing. After around 20 iterations here, the sensor positions become 'good' enough that the regularisation

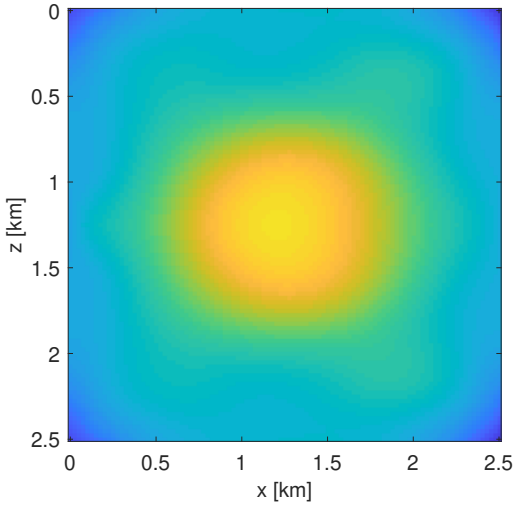
parameter no longer needs to be so large, and so a decrease in α is seen. The several iterations needed for the large increase in α only for α to decrease back to the correct range in the first group seems to show that there is unnecessary work being done in Approach I. In addition, we expect that the large change in α affects the shape of the objective function in the first group and hence the optimal positions converged to at the end of this first group. This in turn determines the final positions of the sensors, as we know from Experiment 2 that after the first group the sensors only make smaller movements. (We see that the change in α affects the final sensor position by comparing the optimal positions found in each approach, i.e., subfigure (a) in Figures 5.1.13, 5.1.14 and 5.1.15).

Figure 5.1.14 shows the results for **Approach II**. Since α is not introduced in the first frequency group, the number of iterations for this group is the same as in Experiment 2, but there is a large increase in the number of iterations in the following frequency groups. Subfigure (c) shows the value of α at each iteration once it is introduced as an optimisation variable (i.e., group 1 is excluded because α is constant there). We see that α does not have the large variation seen in Approach I. It converges to 3.64.

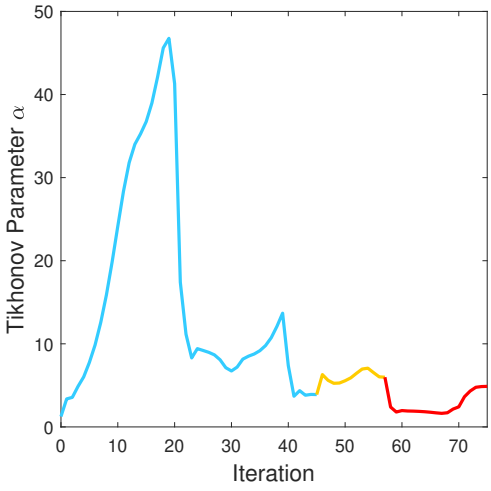
Figure 5.1.15 shows the results for **Approach III**. We see that the optimal sensor positions and FWI reconstruction for Approach II and III are almost exactly the same. Since the optimisation of α occurs in the final frequency group only, there is a large increase in the number of iterations in this final group. Subfigure (c) shows that it takes 45 iterations to optimise α here, compared to Experiment 2 (optimising sensors only) when only 2 iterations were required in the final group. There is also a large decrease in the value of ψ seen in this final group (Subfigure (d)) when compared to the small decrease in the final group seen in Experiment 2 (Figure 5.1.10). We note that the optimal value of α found with Approach III was 3.64.



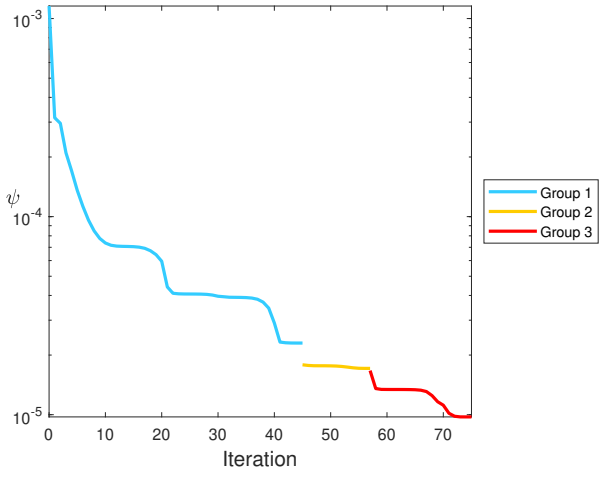
(a) Optimised Sensor Positions $\boldsymbol{p}_{\min}$



(b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min})$

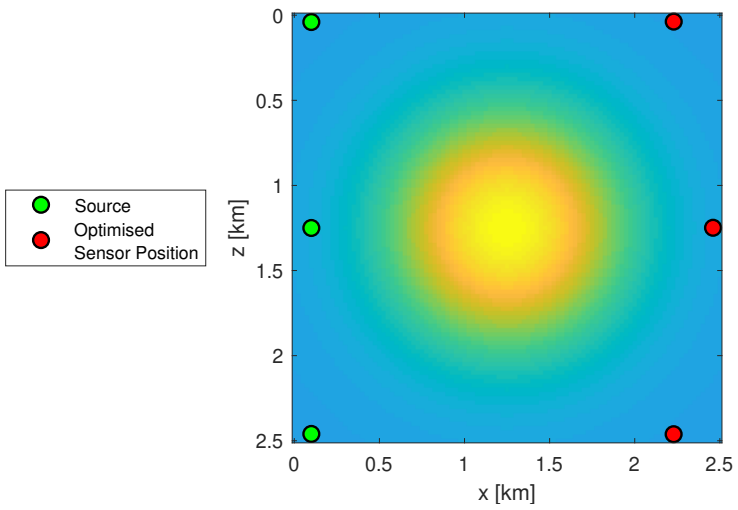


(c) Tikhonov parameter α versus iteration

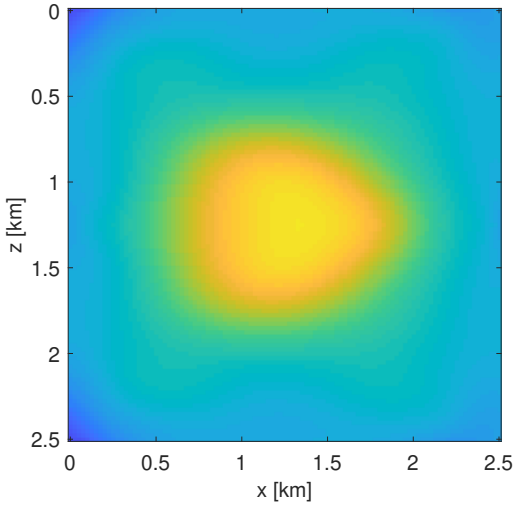


(d) Objective function ψ versus iteration

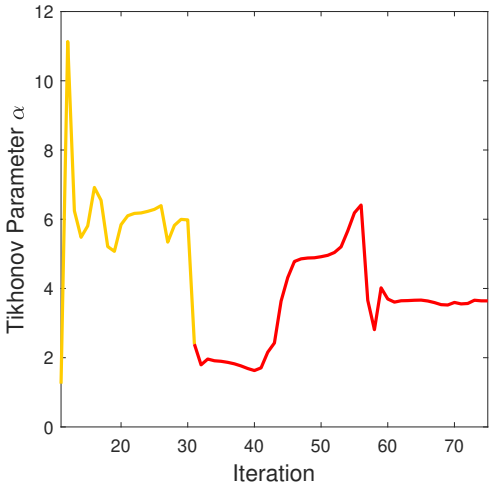
Figure 5.1.13: *Approach I results*



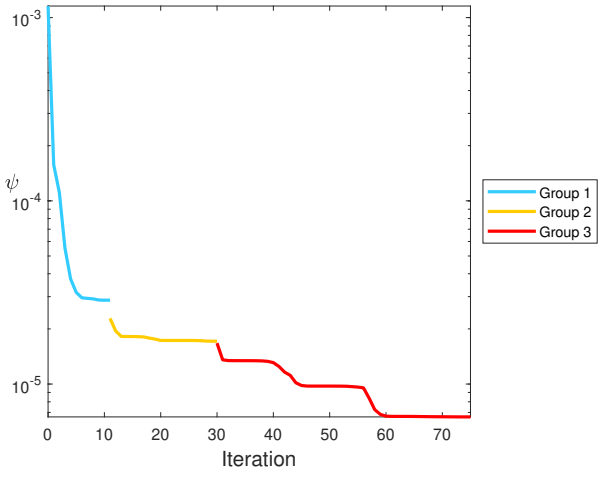
(a) Optimised Sensor Positions $\boldsymbol{p}_{\min}$



(b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min})$

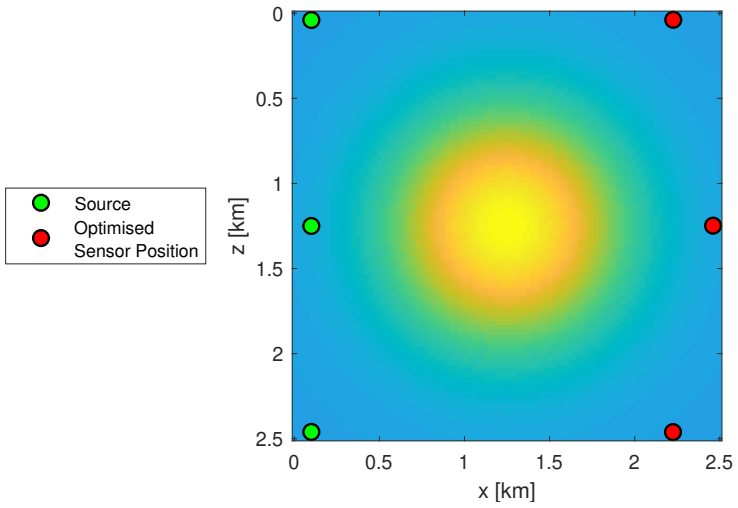


(c) Tikhonov parameter α versus iteration

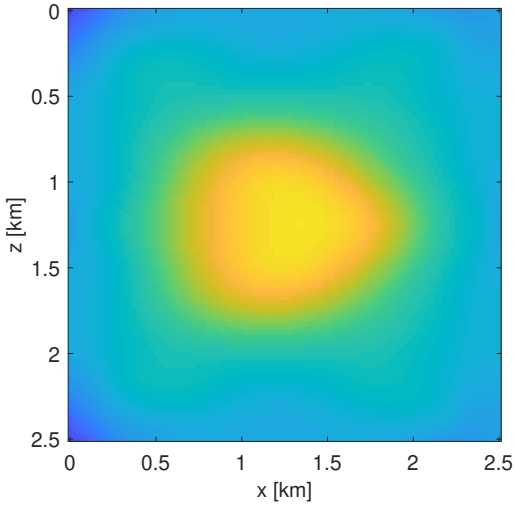


(d) Objective function ψ versus iteration

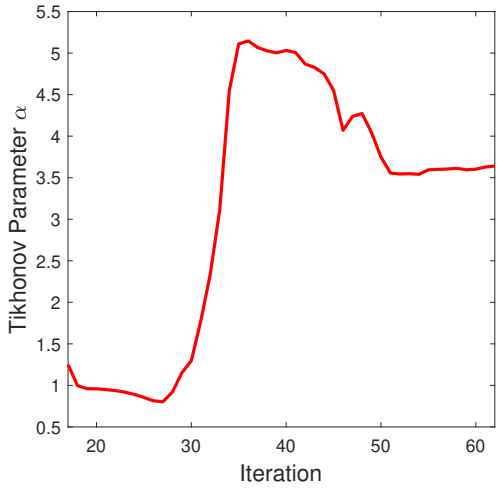
Figure 5.1.14: *Approach II results*



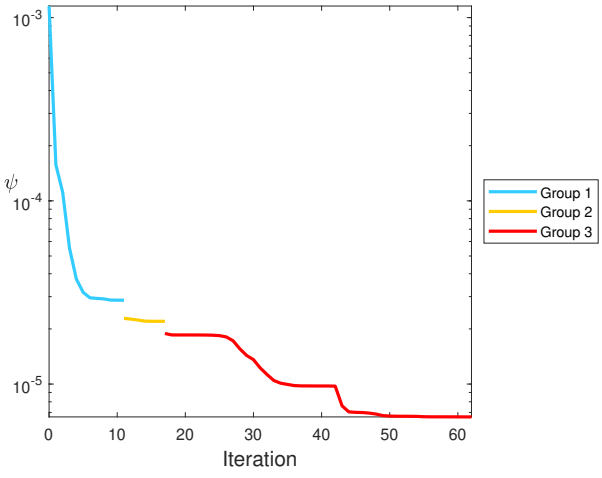
(a) Optimised Sensor Positions $\boldsymbol{p}_{\min}$



(b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min})$



(c) Tikhonov parameter α versus iteration



(d) Objective function ψ versus iteration

Figure 5.1.15: *Approach III results*

Comparison of the Approaches: Table 5.1.1 and Figures 5.1.17 and 5.1.16 provide a direct comparison of these results. In Table 5.1.1 we see that Approach II and III provide the same improvement factors and optimal α's, but that Approach III takes fewer overall iterations to reach this result. We also see that Approach I gives a different improvement factor and optimal α as the other two approaches, and takes the same number of overall iterations as Approach II.

The difference in results between Approach I and Approaches II and III, and the similarity in the latter two approaches, is a consequence of the fact that the first frequency group in the bilevel frequency continuation algorithm has the most influence over the final positions of the sensors, and Approaches II and III have the same value of α in the first group. This can be seen clearly in Figure 5.1.16, where the values of ψ for Approaches II and III coincide for the first group (first 11 iterations). Although they differ after the first group, they eventually converge to the same value. Approach I differs completely from the other two approaches. The comparison of α versus iteration in Figure 5.1.17 shows how large the variation of α is in Approach I, and also shows how quickly Approach III can reach the same/a similar optimal value of α compared to the other two approaches.

Based on these results, Approach III seems to be the best choice in terms of number of iterations. However, as all but one frequency group depends on initial guess of α, if the initial guess for α is poor, this could affect the positions that the sensors converge to, and the sensors may end up in a non-optimal setup. We note that although in this specific example the final improvement factor for Approach I is not as high as that for Approach III, that this is not always the case. In particular, in the case where we have a poor initial guess for α, Approach I will begin to work on correcting this value immediately, and is expected to provide a better result than Approach III. Therefore, if we believe we have a reasonable initial guess for α, then Approach III is the recommended approach, and if we do not have knowledge of reasonable guess for α, Approach I is recommended. However, it is of course acceptable to start optimising α in any frequency group.

Approach	Iteration Count				Optimal α	Improvement Factor
	Group 1	Group 2	Group 3	Total		
I	45	12	18	75	4.88	32.41
II	11	19	45	75	3.64	47.74
III	11	6	45	62	3.64	47.74

Table 5.1.1: *Comparison of results from each approach (values are rounded to two decimal places).*

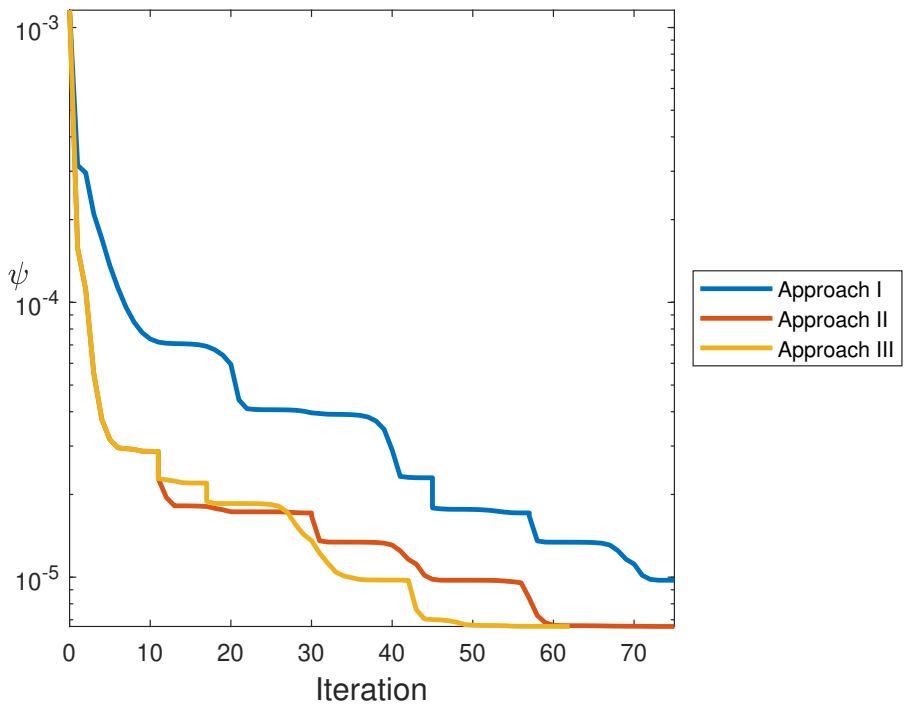


Figure 5.1.16: *Upper-level objective function ψ versus iteration for each approach.*

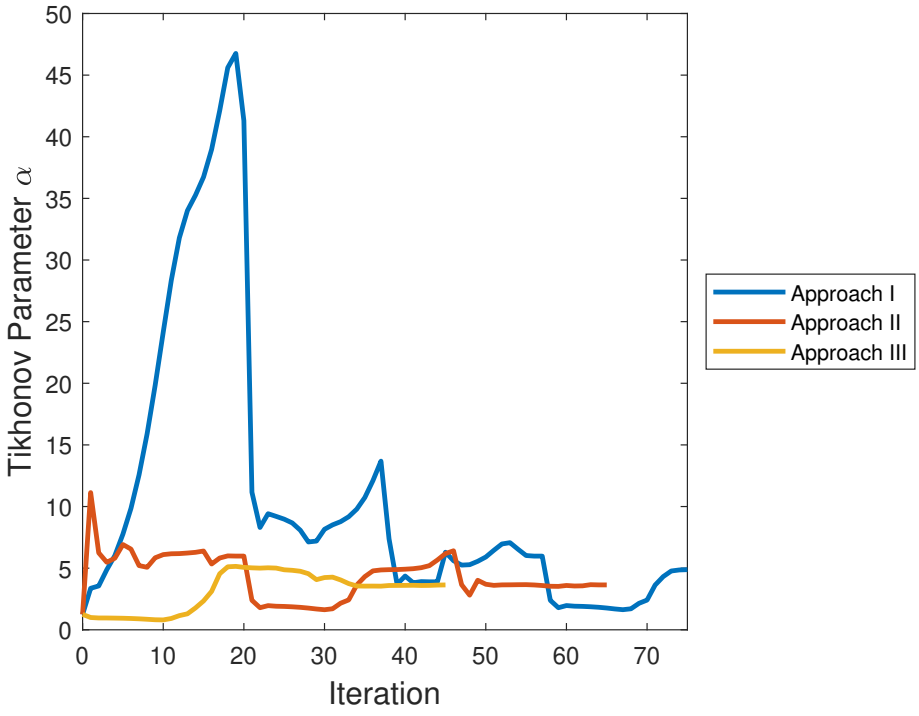


Figure 5.1.17: *Tikhonov regularisation parameter α versus iteration for each approach.*

5.2 Linear System arising in the Upper-Level Gradient

In Section 3.4.2, we derived a formula for the gradient of the upper-level objective function and presented an algorithm for its computation (Algorithm 3.4.9). The computation requires the solution to a linear system involving the FWI Hessian, given by (3.4.20). In this section we discuss solving this linear system.

The system (3.4.20) must be solved for every training model during the computation of the upper-level gradient, and so it is solved many times during the bilevel algorithm. The overall cost of the bilevel algorithm in terms of the number of PDE solves, given by (3.4.18), depends on the number of iterations (N_i) taken to solve (3.4.20). Therefore, it is important that we understand what factors influence the size of N_i and how we can reduce it.

In this section, we solve the system using the conjugate gradient method, and record the number of iterations it takes to converge for various discretisation grids, frequencies and regularisation parameters. We investigate the relationship between the number of iterations needed for convergence, the number of model parameters, the frequency of the problem, the regularisation parameters and the condition number of the Hessian. We also present two novel strategies for preconditioning to reduce the number of iterations to convergence and discuss the cost of each.

We restate the linear system (3.4.20) here,

$$H(\boldsymbol{m}^{FWI}(\boldsymbol{p},\boldsymbol{m}'),\boldsymbol{p})\boldsymbol{\delta}(\boldsymbol{m}^{FWI}(\boldsymbol{p},\boldsymbol{m}'),\boldsymbol{m}') = \boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p},\boldsymbol{m}'), \tag{5.2.1}$$

where the Hessian has the structure

$$H(\boldsymbol{m},\boldsymbol{p}) = H^{(1)}(\boldsymbol{m},\boldsymbol{p}) + H^{(2)}(\boldsymbol{m},\boldsymbol{p}) + \alpha D^T D + \mu I, \tag{5.2.2}$$

where $H^{(1)}$ is defined in (2.4.13), $H^{(2)}$ is defined in (2.4.14), the term $\alpha D^T D$ is the Hessian of the Tikhonov term and αI is the Hessian of the convex term. Section 2.4.3 contains a detailed discussion of the structure of the Hessian and Section 2.4.4 contains a proof of the conditions under which the Hessian is positive definite. We can use the conjugate gradient (CG) method to solve (5.2.1) because the Hessian is symmetric and, by the theory in Section 2.4.4, is positive definite when μ is large enough. In practice we find that the Hessian is positive definite even for small values of μ. We suspect that this is because the Tikhonov parameter helps the positive definitiveness of H, but we don't have a proof of this as the Tikhonov term is only positive semi-definite. Therefore, we still must be cautious and so we always include both regularisation terms, as the $H^{(1)} + H^{(2)}$ term alone is generally indefinite. We note that CG has been used to succesfully solve systems involving the FWI Hessian (evaluated at intermediate models during FWI) in [129].

This section is made up of two subsections - one that discusses the number of iterations taken to solve (3.4.20) with the conjugate gradient method, and the other is concerned with how to reduce this number with preconditioning.

5.2.1 The Non-Preconditioned System

In this section we solve (5.2.1) using the conjugate gradient (CG) method and record the number of iterations, N_i, taken to convergence. We solve different instances of the system obtained by varying the regularisation parameters, α and μ, the size of the discretisation grid, and the frequency. In this section we consider the CG method to have converged if $||\mathbf{r}_n||_2/||\mathbf{r}_0||_2 \leq 10^{-6}$, where $\mathbf{r}_n$ denotes the residual at the nth iteration and $\mathbf{r}_0$ denotes the initial residual. We denote the size of the discretisation grid as $n \times n$, where the grid size h is given by $h = L/(n-1)$, where the domain is and $L \times L$ square. The number of model parameters M is equal to n^2 as we have one model parameter at each grid point.

As the system (5.2.1) involves the FWI solution, $\boldsymbol{m}^{FWI}$, each time we solve the system in this section we must first solve the FWI problem. We solve the FWI problem for the training/ground truth model and acquisition setup shown in Figure 5.2.1. We perform FWI using synthetic data, taking care to avoid an inverse crime. Also, even though we don't require the explicit computation of the Hessian to solve (5.2.1) (because we only require Hessian-vector products), we sometimes do compute this separately so that we can record its condition number. We compute the condition number of the Hessian using the `cond` function in Matlab.

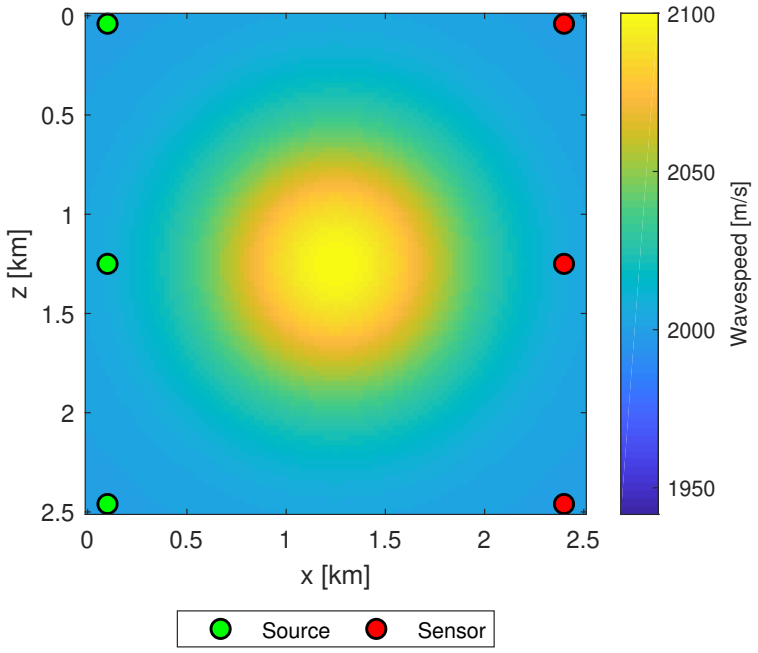


Figure 5.2.1: *Ground truth model and acquisition setup used for experiements in this section.*

Number of Iterations for Varied Regularisation Parameters: Here we vary the Tikhonov parameter α and convex parameter μ, and record the resulting number of CG iterations, N_i, to solve (5.2.1). Varying the regularisation parameter has two main consequences to the linear system (5.2.1). The first is that the FWI reconstruction,

$\boldsymbol{m}^{FWI}$, changes which therefore affects the right-hand side of (5.2.1) directly and affects the system matrix as well, since the Hessian (5.2.2) is evaluated at $\boldsymbol{m}^{FWI}$. The second consequence is that the regularisation terms in the Hessian (5.2.2) vary. Therefore, along with the number of iterations, we also record ψ (to measure the quality of the reconstruction $\boldsymbol{m}^{FWI}$) and the condition number of the Hessian, $\kappa(H)$. (Recall that ψ is the half 2-norm squared of the right-hand side of (5.2.1).) We will see that, since ψ and $\kappa(H)$ are affected differently by the regularisation parameters, it can be difficult to see any relationship between the regularisation parameters and the number of iterations.

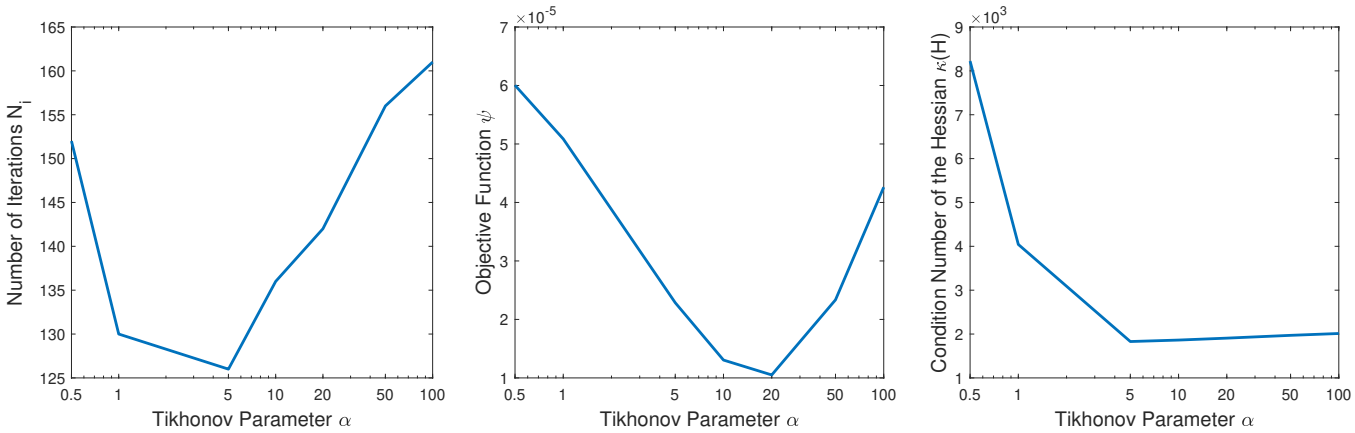
The computations here are performed on a 50×50 grid. The frequency is constant at 2 Hz (which equates to approximately 2.5 wavelengths in the domain). Note that when α is being varied, μ is kept constant and vice versa. We vary α and μ in appropriate ranges such that a reasonable reconstruction is found. For every variation of regularisation parameters here, the Hessian was found to be positive definite.

α	N_i	ψ	$\kappa(H)$
0.5	152	6.0034×10^{-5}	8.2230×10^{3}
1	130	5.0861×10^{-5}	4.0430×10^{3}
5	126	2.2854×10^{-5}	1.8294×10^{3}
10	136	1.3052×10^{-5}	1.8628×10^{3}
20	142	1.0507×10^{-5}	1.9068×10^{3}
50	156	2.3348×10^{-5}	1.9705×10^{3}
100	161	4.2605×10^{-5}	2.0126×10^{3}

Table 5.2.1: *Effect of varying Tikhonov regularisation parameter α on solving (5.2.1). The convex parameter is constant at $\mu = 10^{-8}$.*

Table 5.2.1 shows the effect of varying the Tikhonov regularisation parameter α on solving (5.2.1). The number of iterations is clearly impacted by the value of α, but the relationship between α and N_i is not simple. As α is increased from 0.5, N_i decreases until we reach $\alpha = 10$, where N_i increases again. In Figure 5.2.2, we see that the relationship between the Tikhonov parameter and the number of iterations N_i to solve the system (5.2.1) is complicated, and seems to be affected by the combination of the quality of the reconstruction $\boldsymbol{m}^{FWI}$ and the condition number of the Hessian. We also see that N_i does not blow up across the range of α.

In Table 5.2.2 we study the effect of varying the convex regularisation parameter μ. The convex parameter μ appears to affect N_i less than α. The number of iterations is constant as μ is increased from 10^{-10} to 10^{-6}. The error in the reconstruction ψ and the condition number of the Hessian $\kappa(H)$, varies by only a very small amount up to $\mu = 10^{-7}$, and has a slightly larger change at $\mu = 10^{-6}$. When $\mu = 10^{-5}$, the error in the reconstruction increases and $\kappa(H)$ reduces substantially. It is only at this point that we see a decrease in the number of iterations. However, the reconstruction becomes very poor when μ is increased too much, and so a large value of μ would not be used in practice. Therefore, in practice, the chosen value of μ does not appear to affect the number of iterations taken to solve (5.2.1).



(a) Number of Iterations N_i (b) Objective Function ψ (c) Condition Number of Hessian $\kappa(H)$

Figure 5.2.2: *Plots of values in Table 5.2.1 showing the effect of varying Tikhonov parameter α. Note that the x-axis is on a log scale so that the variation in α is noticeable.*

μ	N_i	ψ	$\kappa(H)$
10^{-10}	126	2.2864×10^{-5}	1.8331×10^{3}
10^{-9}	126	2.2862×10^{-5}	1.8327×10^{3}
10^{-8}	126	2.2854×10^{-5}	1.8294×10^{3}
10^{-7}	126	2.2811×10^{-5}	1.7969×10^{3}
10^{-6}	126	2.7623×10^{-5}	1.5259×10^{3}
10^{-5}	110	4.4471×10^{-4}	6.1756×10^{2}

Table 5.2.2: *Effect of varying the convex regularisation parameter μ on solving (5.2.1). The Tikhonov parameter is constant at $\alpha = 5$.*

Number of Iterations for Varied Grid Size: In this experiment, we vary the grid size that the domain is discretised on, i.e., we vary the mesh diameter $h = L/(n-1)$, and consequently the number of discretisation nodes in each direction n, and the number of model parameters M. We choose a constant frequency, 0.5 Hz, that is low enough to ensure numerical accuracy for the coarsest grid. On the coarsest grid there are at least 32 grid points per wavelength, and on the finest grid there are at least 200 grid points per wavelength. We choose the combination of regularisation parameters $\alpha = 5$ and $\mu = 10^{-6}$ on the 50×50 grid, chosen because this combination produces a good quality reconstruction. To ensure we are solving the same FWI problem on all grids (i.e., that the balance of the misfit and two regularisation terms remains the same and that equivalent reconstructions, $\boldsymbol{m}^{FWI}$, are produced) the regularisation parameters are scaled with the grid size. Each time the linear system (5.2.1) is solved, we record the number of iterations N_i and the condition number of the Hessian $\kappa(H)$.

$n \times n$	M	N_i	$\kappa(H)$
21×21	441	52	0.2143×10^3
26×26	676	65	0.3266×10^3
51×51	2601	132	1.2599×10^3
101×101	10201	265	4.9319×10^3
126×126	15876	330	7.6746×10^3

Table 5.2.3: *Effect of varying the grid size $n \times n$ on solving (5.2.1).*

Table 5.2.3 shows that as the discretisation grid is made finer, i.e., as n is increased, the number of iterations N_i and the condition number of the Hessian $\kappa(H)$ both increase as well. By performing extrapolation on the data in Table 5.2.3, we have the following approximate relationships,

$$N_i \propto n, \tag{5.2.3}$$
$$\kappa(H) \propto M = n^2, \tag{5.2.4}$$

where $\propto$ denotes proportionality. We also find the following relationship,

$$N_i \propto \sqrt{\kappa(H)}. \tag{5.2.5}$$

This observed behaviour agrees with that noted in the literature (for example, see [183, Section 3]).

Number of Iterations for Varied Frequency: In this experiment, we investigate how the number of iterations is affected by frequency f. We keep the product $h\omega$ constant (which keeps the number of grid points per wavelength constant). We recall that $h = L/(n-1)$ and $\omega = 2\pi f$.

$n \times n$	f	N_i	$\kappa(H)$
26×26	0.5	65	0.3267×10^3
51×51	1	129	1.6145×10^3
101×101	2	258	6.0978×10^3
201×201	4	532	28.633×10^3

Table 5.2.4: *Effect of varying the frequency (while keeping $h\omega$ constant) on solving (5.2.1).*

In Table 5.2.4, we can see that as frequency is doubled, the number of iterations is approximately doubled. We can also see that the condition number of the Hessian increases as the frequency increases. Extrapolating the data in Table 5.2.4, we see the

following power-law relationships,

$$N_i \propto \omega, \tag{5.2.6}$$
$$\kappa(H) \propto \omega^2. \tag{5.2.7}$$

These relationships agree with the previous section that $N_i \propto \sqrt{\kappa(H)}$. The results (5.2.6) and (5.2.7) are a manifestation of the fact that $N \sim f$ here.

5.2.2 Preconditioners

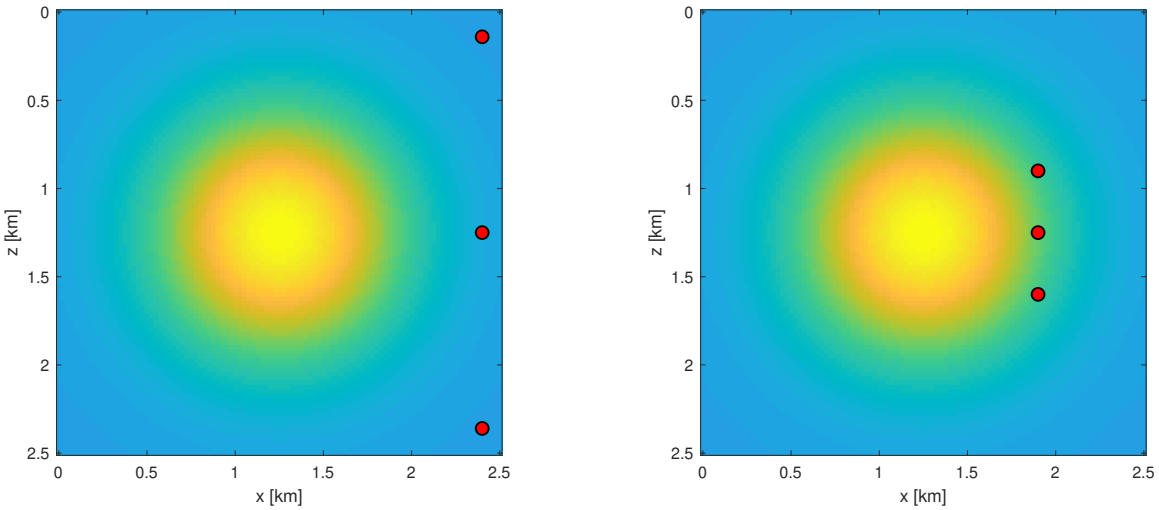
As seen in Section 5.2.1, the number of iterations taken to solve (5.2.1) with the CG method can be quite large, and increases when the number of model parameters is increased. This system must be solved many times during the bilevel algorithm - it is solved every time the gradient is computed and for each training model. Therefore, it would be helpful to reduce the number of iterations as much as possible. To do so, we consider the preconditioned conjugate gradient (PCG) method. We propose the following two ideas for preconditioners:

- **Preconditioner 1:**
$$P_1 = H^{-1}(\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \boldsymbol{m}'), \boldsymbol{p}_0)$$
This preconditioner involves computing the full Hessian explicitly at the initial guess $\boldsymbol{p}_0$. The inverse of this Hessian is then used as the preconditioner for (5.2.1). During the bilevel algorithm the sensor positions $\boldsymbol{p}$ will move away from the initial guess $\boldsymbol{p}_0$, and therefore the reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')$ will change from $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \boldsymbol{m}')$. We will see later that the effectiveness of this preconditioner is reduced for $\boldsymbol{p}$ not near $\boldsymbol{p}_0$. Therefore, the preconditioner may have to be recomputed as the bilevel algorithm progresses to ensure the preconditioner is still effective. Since the Hessian involves a sum over frequency, the strategy we propose is to recompute the preconditioner at the beginning of each frequency group. We note that since the Hessian depends on $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \boldsymbol{m}')$, which is the reconstruction of the training model $\boldsymbol{m}'$, the preconditioner will be different for each training model. The cost of computing this preconditioner, in terms of the number of PDE solves, is M PDE solves, for each source, frequency and training model.

- **Preconditioner 2:**
$$P_2 = H_{reg}^{-1} = \left(\alpha D^T D + \mu I\right)^{-1}$$
This preconditioner involves computing the regularisation terms of the Hessian. This is cheap to compute as no PDE solves are required. In addition, this preconditioner is independent of sensor position $\boldsymbol{p}$, FWI model $\boldsymbol{m}^{FWI}$, training models $\boldsymbol{m}'$ and frequency, so when optimising sensor positions alone, it only needs to be computed once at the beginning of the bilevel algorithm. This section mainly focuses on optimising sensors alone but we show later that even when optimising α, this preconditioner does not need to be recomputed since it remains effective even when α is no longer near its initial guess.

We note that although we write the preconditioners as the inverse of matrices in the formulae above, we do not invert the full matrices directly in practice. Instead, we compute the Cholesky factorization of these matrices (as both preconditioners should be positive definite) and store and invert the resulting lower triangular matrix.

We now do some experiments to demonstrate the reduction in the number of iterations for PCG to converge, compared to the number of CG iterations in Section 5.2.1. We recall that we solve (5.2.1) for the problem shown in Figure 5.2.1. We denote the number of iterations taken using Preconditioner 1 as $N_i^{P_1}$ and the number of iterations taken using Preconditioner 2 as $N_i^{P_2}$. As we have explained, the preconditioner P_1 depends on the sensor positions. Therefore we test two versions of P_1 - one where the sensor positions $\boldsymbol{p}_0$ are close to the current sensor positions $\boldsymbol{p}$ (i.e., close to those shown in Figure 5.2.1) and one where the sensor positions $\boldsymbol{p}_0$ are far from $\boldsymbol{p}$. We denote these preconditioners as $P_{1_{near}}$ and $P_{1_{far}}$ and their iterations counts as $N_i^{P_{1near}}$ and $N_i^{P_{1far}}$ respectively. We display these 'near' and 'far' sensor setups in Figure 5.2.3 (a) and (b) respectively.



(a) Sensor positions 'near' $\boldsymbol{p}$ (b) Sensor positions 'far' from $\boldsymbol{p}$

Figure 5.2.3: *Sensor positions used to compute the preconditioner P_1. This preconditioner is used to solve (5.2.1) for the problem shown in Figure 5.2.1.*

We now repeat some of the experiments in Section 5.2.1 with these preconditioners.

Number of Iterations for Varied Regularisation Parameters: Table 5.2.5 and Figure 5.2.4 show the reduction in iterations needed to solve (5.2.1) when using the PCG method. We see that preconditioner P_1 is very effective at reducing the number of iterations when the sensors $\boldsymbol{p}_0$ are close to $\boldsymbol{p}$. The number of iterations are reduced by between 85-96%. When $\boldsymbol{p}_0$ is not close to $\boldsymbol{p}$ however, P_1 is not as effective. In this case, the PCG method is even worse than the CG method when α is small, but improves as α is increased, reaching approximately a 89% reduction in number of iterations at best. The preconditioner P_2 produces a more consistent reduction in the number of iterations, ranging from 71-91%.

α	N_i	$N_i^{P_{1near}}$	$N_i^{P_{1far}}$	$N_i^{P_2}$
0.5	152	24	180	43
1	130	19	136	36
5	126	11	64	33
10	136	10	45	26
20	142	8	33	22
50	156	7	24	16
100	161	6	17	14

Table 5.2.5: *Effect of varying Tikhonov regularisation parameter α on solving (5.2.1) using PCG. The convex parameter is constant at $\mu = 10^{-8}$.*

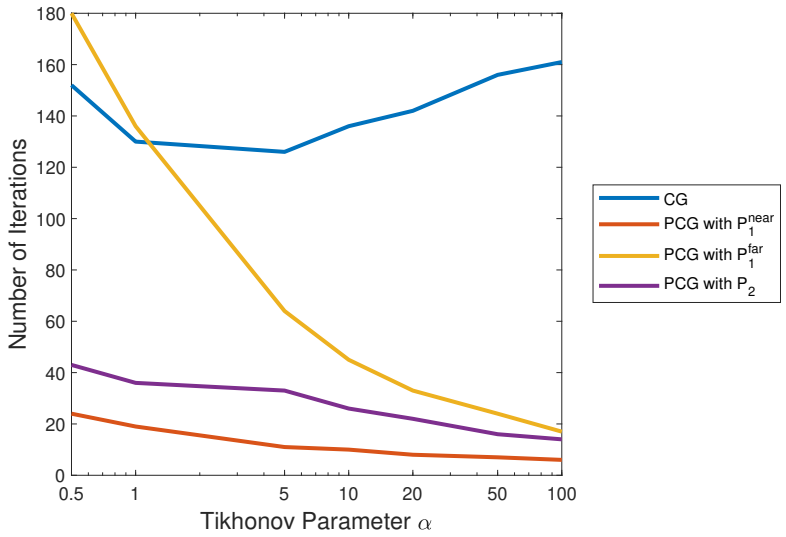
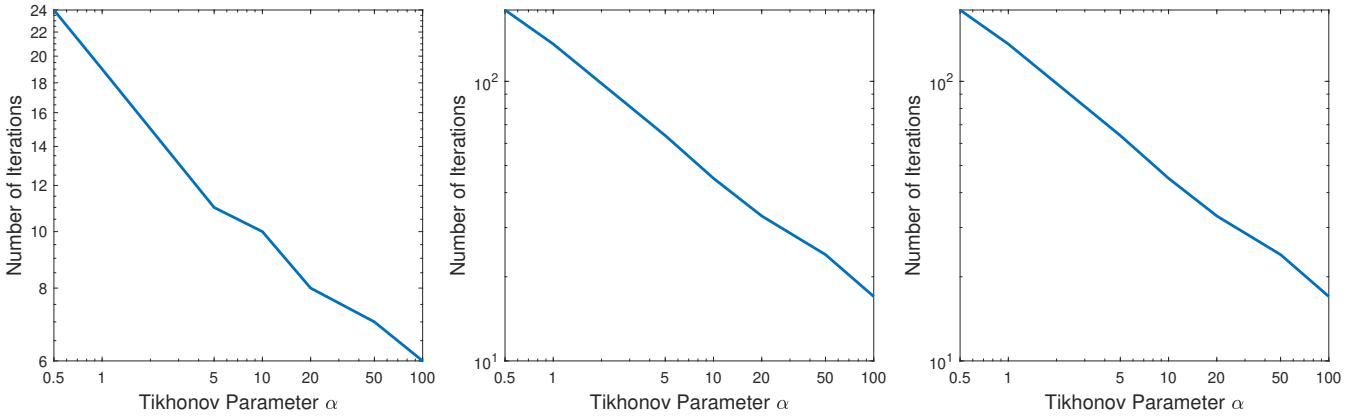


Figure 5.2.4: *Plot of values in Table 5.2.5, showing the effect of varying the Tikhonov parameter on solving (5.2.1) using PCG compared to CG. Note that the x-axis is on a log scale so that the variation in α is noticeable.*

Figure 5.2.5 shows the relationship between α and the PCG iterations more clearly by plotting on a log-log scale. While the number of CG iterations has a complicated relationship with α, the relationship with the number of PCG iterations is much more straightforward, and an increase in α produces a decrease in the number of PCG iterations. The decrease is fast for $N_i^{P_{1far}}$ (subfigure (b)), and we observe an approximate relationship like $N_i^{P_{1far}} \propto \alpha^{-0.45}$. For the more effective P_1 preconditioner (subfigure (a)), we observe an approximate relationship like $N_i^{P_{1near}} \propto \alpha^{-0.26}$. The slowest decrease is observed for P_2 (subfigure (c)), and we find the approximate relationship $N_i^{P_2} \propto \alpha^{-0.21}$.



(a) $N_i^{P_{1near}}$ (b) $N_i^{P_{1far}}$ (c) $N_i^{P_2}$

Figure 5.2.5: *Log-log plots of the number of PCG iterations versus the Tikhonov parameter α.*

Table 5.2.6 and Figure 5.2.6 demonstrate how the number of iterations vary with the convex parameter μ. We have seen that the number of CG iterations generally stays constant with μ, and decreases when μ becomes too large and produces a poor reconstruction (at $\mu = 10^{-5}$). We do not see the this type of behaviour with P_1, and there is an increase in the number of iterations when the reconstruction is poor. The number $N_i^{P_{1near}}$ remains constant, at approximately a 90% reduction, until $\mu = 10^{-5}$ where the iteration count increases by one. The number $N_i^{P_{1far}}$ varies slightly until $\mu = 10^{-6}$, when there is a larger increase in the number of iterations. The behaviour for P_2 is completely different, and we see a steady decrease in $N_i^{P_2}$ as μ is increased. In conclusion, the behaviour of the two preconditioners is affected very differently by a change in μ.

μ	N_i	$N_i^{P_{1near}}$	$N_i^{P_{1far}}$	$N_i^{P_2}$
10^{-10}	126	11	63	37
10^{-9}	126	11	63	35
10^{-8}	126	11	64	33
10^{-7}	126	11	62	31
10^{-6}	126	11	68	30
10^{-5}	110	12	71	27

Table 5.2.6: *Effect of varying the convex regularisation parameter μ on solving (5.2.1) using PCG. The Tikhonov parameter is constant at $\alpha = 5$.*

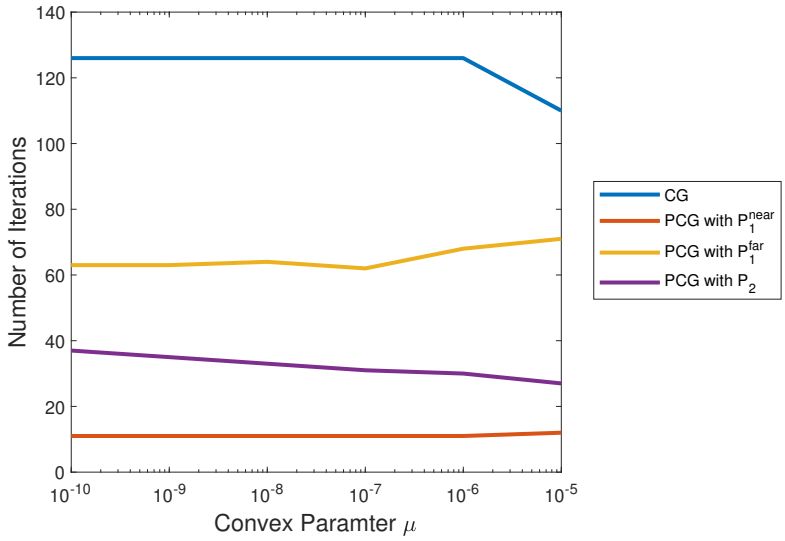


Figure 5.2.6: *Plot of values in Table 5.2.6, showing the effect of varying the convex parameter on solving (5.2.1) using PCG. Note that the x-axis is on a log scale so that the variation in μ is noticeable.*

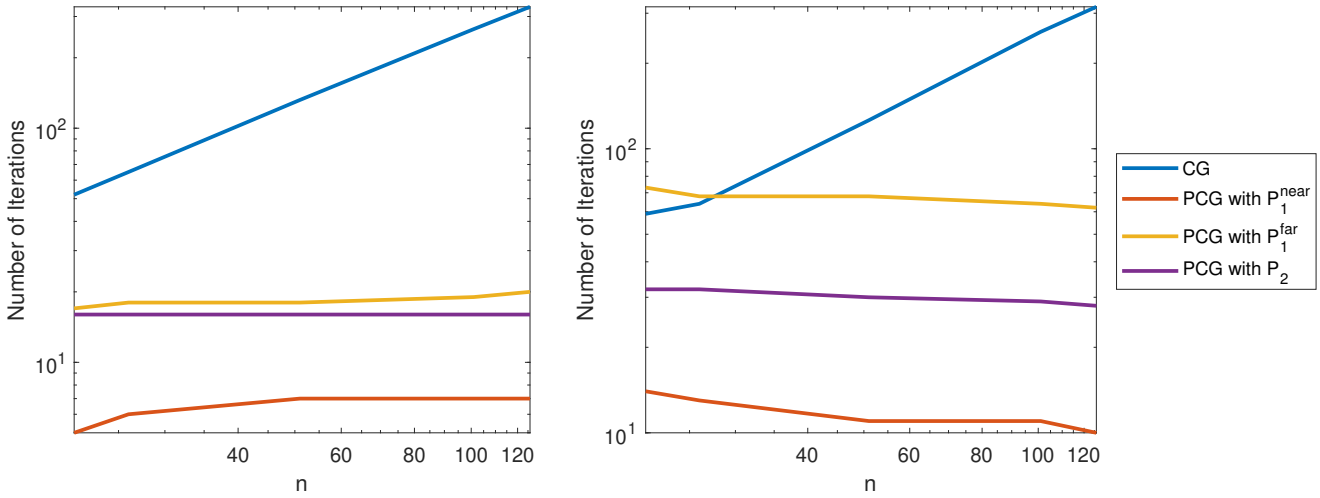
Number of Iterations for Varied Grid Size: As in Section 5.2.1, we vary the grid size that the domain is discretised on and record the resulting number of iterations to solve (5.2.1). We repeat this for two frequencies, 0.5 Hz in Table 5.2.7, and 2 Hz in Table 5.2.8. For the lower frequency, the PCG iterations using P_1 grows very slowly as the grid is refined, but for the higher frequency the number of iterations decrease as the grid is refined. For the lower frequency, the PCG iterations using P_2 stays constant as grid size is refined, while for the higher frequency the number of iterations decreases. This behaviour is likely related to the fact that as frequency increases, we require a finer mesh to avoid larger numerical errors. Therefore for the 2 Hz table, we expect that the finer mesh results in a more accurate FWI reconstruction, and hence less iterations are required to solve (5.2.1) to the required tolerance. For the 0.5 Hz problem, the frequency is so low that we don't observe this effect. For both frequencies, using PCG with either preconditioner solves the issue we had with the CG method where the number of iterations grows with grid size. This behaviour is seen clearly in Figure 5.2.7.

$n \times n$	M	N_i	$N_i^{P_{1near}}$	$N_i^{P_{1far}}$	$N_i^{P_2}$
21 × 21	441	52	5	17	16
26 × 26	676	65	6	18	16
51 × 51	2601	132	7	18	16
101 × 101	10201	265	7	19	16
126 ×126	15876	330	7	20	16

Table 5.2.7: *Effect of varying the grid size $n \times n$ on solving (5.2.1). Frequency is constant at 0.5 Hz.*

$n \times n$	M	N_i	$N_i^{P_{1near}}$	$N_i^{P_{1far}}$	$N_i^{P_2}$
21 × 21	441	59	14	73	32
26 × 26	676	64	13	68	32
51 × 51	2601	126	11	68	30
101 × 101	10201	258	11	64	29
126 ×126	15876	316	10	62	28

Table 5.2.8: *Effect of varying the grid size $n \times n$ on solving (5.2.1). Frequency is constant at 2 Hz.*



(a) 0.5 Hz (b) 2 Hz

Figure 5.2.7: *Log-log plots of values in Tables 5.2.7 and 5.2.8 showing the effect of varying grid size $n \times n$ on the number of iterations for the PCG method with different preconditioners compared to the CG method.*

Number of Iterations for Varied Frequency: In Table 5.2.9, we record the number of PCG iterations for varying frequency, while keeping the product $h\omega$ constant. For both P_1 and P_2, the number of iterations increase with frequency, as shown in Figure 5.2.8. We observe the following approximate relationships: $N_i^{P_{1near}} \propto \omega^{0.44}$, $N_i^{P_{1far}} \propto \omega^{0.88}$, and $N_i^{P_2} \propto \omega^{0.39}$. Therefore we see that the number of PCG iterations with preconditioner P_1 grows faster with frequency than the number of PCG iterations with preconditioner P_2.

$n \times n$	f	N_i	$N_i^{P_{1near}}$	$N_i^{P_{1far}}$	$N_i^{P_2}$
26×26	0.5	65	6	18	16
51×51	1	129	8	29	21
101×101	2	258	11	62	28
201×201	4	532	15	106	36

Table 5.2.9: *Effect of varying the frequency (while keeping $h\omega$ constant) on solving (5.2.1).*

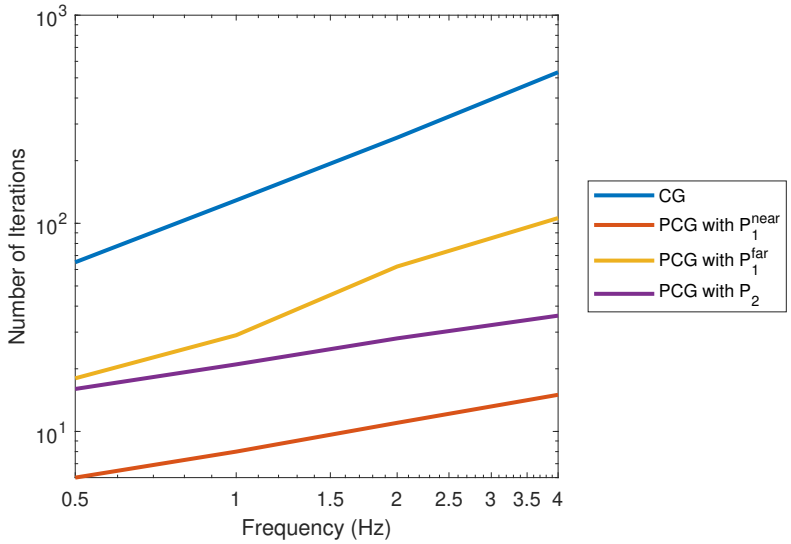


Figure 5.2.8: *Log-log plots of values in Table 5.2.9 showing the effect of varying frequency on the number of iterations for the PCG method with different preconditioners compared to the CG method.*

Optimising α: So far in this section we have only considered the case where sensor positions alone are being optimised. We did this by testing the preconditioner P_1 at different sensor positions, and observed that P_1 becomes less effective when the sensor positions at which P_1 is computed is far from the sensor positions in the system (5.2.1). This observed behaviour led to the conclusion that P_1 should be recomputed throughout the bilevel algorithm. Here we consider the case where α is also being optimised. Both preconditioners, P_1 and P_2, are computed using the initial α, and so we need to test the effectiveness of P_1 and P_2 as α is varied from its initial guess. As we already have a strategy for recomputing P_1, we use the results here to decide whether/how often P_2 should be recomputed as α is varied in the bilevel algorithm.

When α is being optimised, the linear system (5.2.1) is rewritten as a function of α,

$$H(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha,\boldsymbol{m}'),\boldsymbol{p},\alpha)\boldsymbol{\delta}(\boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha,\boldsymbol{m}'),\boldsymbol{m}') = \boldsymbol{m}' - \boldsymbol{m}^{FWI}(\boldsymbol{p},\alpha,\boldsymbol{m}'). \quad (5.2.8)$$

The experiment here involves considering an initial α at which the preconditioners are computed, and then varying α in (5.2.8) (which involves performing FWI for the new α to get $\boldsymbol{m}^{FWI}$). We record the number of iterations as α is varied. We choose an initial $\alpha = 5$, and compute the preconditioners at this value. The α in (5.2.8) is varied according to the first column of Table 5.2.10. Therefore the row where $\alpha = 5$ corresponds to that same row in Table 5.2.5 (i.e., the case where α has not been changed after the preconditioner is computed). All other rows are compared to this row. We note that for these experiments, we use a 50×50 grid, the frequency is kept constant at 2 Hz and the convex parameter is kept constant at 10^{-8}.

α	N_i	$N_i^{P_1 near}$	$N_i^{P_1 far}$	$N_i^{P_2}$
0.5	152	24	103	46
1	130	19	90	38
5	126	11	64	33
10	136	13	54	25
20	142	15	51	21
50	156	20	46	15
100	161	24	42	13

Table 5.2.10: *Effect of α in (5.2.8) being varied but the preconditioner staying constant at $\alpha = 5$. The convex parameter is constant at $\mu = 10^{-8}$.*

The iteration counts in Table 5.2.10 show that as α moves away from $\alpha = 5$, $N_i^{P_1 near}$ gets larger, however in general it remains small and is still a substantial reduction from N_i. The iteration counts $N_i^{P_1 far}$ have a different behaviour - the iteration counts increase quickly as α is decreased from 5, and the iteration counts decrease as α is increased from 5. This distinct difference in the behaviour of $N_i^{P_1 near}$ and $N_i^{P_1 far}$ demonstrates that the manner in which the change in Tikhonov parameter impacts the effectiveness

of P_1 depends heavily on the sensor positions. In addition, we observe that the change in sensor positions appear to have a more significant effect on the number of iterations, and recomputing P_1 at the beginning of each frequency group still appears to be the best approach to keep the number of iterations low.

The iteration counts for P_2 increase as α is decreased from 5 and they decrease as α is increased from 5. All iteration counts for P_2 are still a large improvement from the unpreconditioned conjugate gradient method. Therefore, although the effectiveness of P_2 will vary depending on how α changes during the bilevel algorithm, this change may be favourable (particularly if the initial guess for α is smaller than its optimal value), and there is not necessarily any advantage gained by recomputing P_2 during the bilevel algorithm. Therefore, we conclude that P_2 (or specifically the Cholesky factor of $\alpha D^T D + \mu I$) just needs to be computed once, at the beginning of the bilevel algorithm.

5.2.2.1 Cost

We discuss here more details about the cost of the sensor optimisation problem where (5.2.1) is solved with CG versus PCG for our two different preconditioners. The cost is measured in terms of the number of PDE solves. We remark that the estimates of cost are given in terms of the number of iterations N_i, $N_i^{P_1}$ and $N_i^{P_2}$. In reality these numbers are not constant and vary throughout the bilevel algorithm, so when we include them in the following cost analysis what we really mean in practice is the average values of N_i, $N_i^{P_1}$ and $N_i^{P_2}$.

We recall that when (5.2.1) is solved with CG, the overall cost of the bilevel algorithm in terms of the number of PDE solves is given by (3.4.18). We note that (3.4.18) is the cost for solving the bilevel problem with gradient descent, and so the factor N_u, which is the number of upper-level iterations, is also just the number of times that the upper-level gradient is computed. We note here that when a line search algorithm is used in the upper-level optimisation method, there is more gradient computations required than the number of upper-level iterations. To account for this in this cost analysis, from here on we will replace N_u by N_u^{ls}, which we refer to as the total number of gradient computations. In the case where a line search algorithm is used N_u^{ls} is the product of the number of upper-level iterations and the average number of line search iterations per upper-level iteration. When no line search algorithm is used, N_u^{ls} is simply equal to N_u. We also note that, assuming the bilevel frequency continuation approach is used, N_ω denotes the total number of frequencies used across all frequency groups, for example if there is a shared frequency between groups then this frequency is counted twice. We rewrite the overall cost of the bilevel when solving (5.2.1) with CG here,

$$\#\ \textbf{PDE Solves when using CG} = \boldsymbol{N_{m'} N_u^{ls} N_s N_\omega\,(2N_l + 2N_i + dN_r)}, \quad (5.2.9)$$

When solving (5.2.1) with PCG using P_1, the factor N_i becomes $N_i^{P_1}$ which, as seen in the tables in this section, is usually a reduction but can vary dramatically during the algorithm. However, there is an additional cost involved in computing this preconditioner. This is the cost of computing a full Hessian, which depends on the number of model parameters M, which is usually very large. Assuming we recompute

the preconditioner at the beginning of each frequency group, the overall cost of the bilevel when solving (5.2.1) with PCG with P_1 is

$$\textbf{\# PDE Solves when using PCG with } \mathbf{P_1} = \boldsymbol{N_{m'} N_s N_\omega \left(N_u^{ls} \left(2N_l + 2N_i^{P_1} + dN_r \right) + M \right)}. \quad (5.2.10)$$

Therefore, for P_1 to reduce the overall cost of the bilevel algorithm, we require the following to be true

$$\left(N_i - N_i^{P_1} \right) > \frac{M}{2N_u^{ls}},$$

i.e., we require the difference in iterations between the CG and PCG method to be greater than some threshold involving the number of model parameters and number of gradient evaluations. To understand how likely this is, we use the example of Table 5.2.7. Consider the problem on the 101×101 grid. The value of $N_i^{P_1}$ varies depending on how far away the sensor positions are from the starting guess so here we estimate $N_i^{P_1}$ as an average of the 'near' and 'far' values, giving the difference $N_i - N_i^{P_1}$ as 252. Therefore, for P_1 to be effective in the bilevel algorithm, there needs to be are least 21 gradient evaluations. Between upper-level iterations and line search evaluations, 21 gradient evaluations is often reached in practice.

When solving (5.2.1) with PCG using P_2, the factor N_i becomes $N_i^{P_2}$, which as seen in the earlier tables is consistently a reduction in the number of iterations. Computing P_2 does not require any additional PDE solves. Therefore, the overall cost of the bilevel when solving (5.2.1) with PCG with P_2 is,

$$\textbf{\# PDE Solves when using PCG with } \mathbf{P_2} = \boldsymbol{N_{m'} N_u^{ls} N_s N_\omega \left(2N_l + 2N_i^{P_2} + dN_r \right)}. \quad (5.2.11)$$

Therefore, for P_2 to reduce the overall cost, we simply require

$$N_i^{P_2} < N_i,$$

which we expect to always be true based on our earlier experimentation.

Comparing the two preconditioning strategies, P_1 results in an overall cheaper algorithm than P_2 when the following is true,

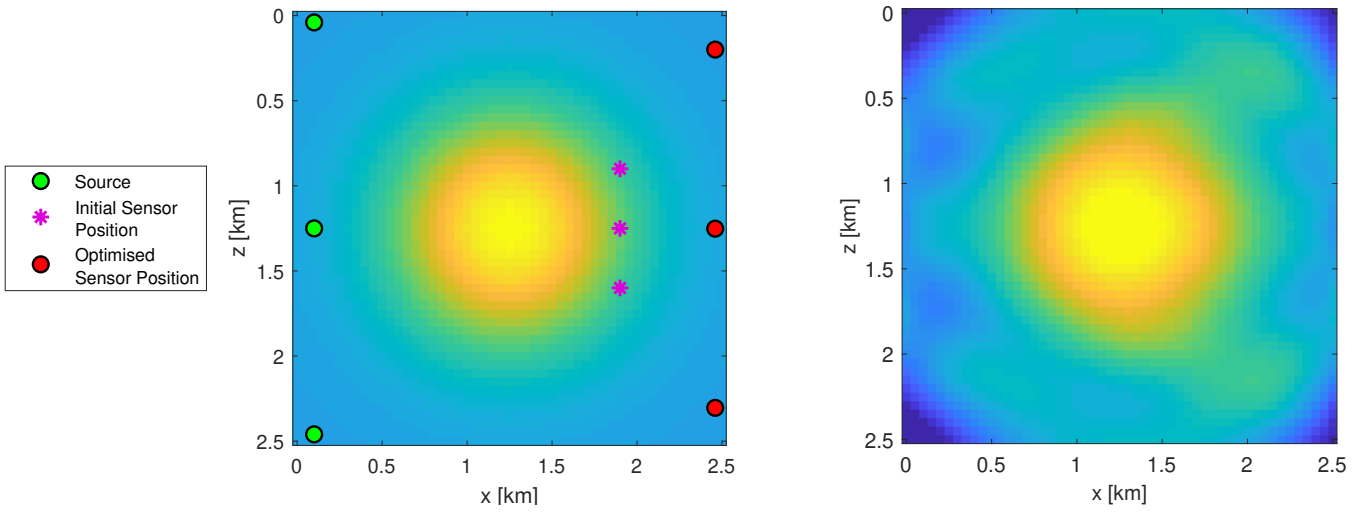
$$\left(N_i^{P_2} - N_i^{P_1} \right) > \frac{M}{2N_u^{ls}}.$$

Since the difference between $N_i^{P_2}$ and $N_i^{P_1}$ can be small, P_1 is usually only more effective than P_2 if the size of the problem, M, is small enough, or there are many gradient evaluations N_u^{ls}. For problems with a large M, we would expect P_2 to be more effective than P_1. As an example, we return to the 101×101 grid row in Table 5.2.7. Considering $N_i^{P_1}$ as an average of the 'near' and 'far' values, the difference $N_i^{P_2} - N_i^{P_1}$ is 3. By the above inequality, P_2 would be more effective in this case. The preconditioner P_1 would only be more effective if the number of gradient evaluations became greater than 1700, which should not generally occur in practice.

5.2.2.2 Experiments

We now include some full sensor optimisation examples to investigate the overall speedup achieved through the use of preconditioners P_1 and P_2 in practice. Each sensor optimisation example is repeated three times - once without a preconditioner, once using P_1 and once using P_2. We report the time taken for the bilevel algorithm to finish in each case. The aim of these experiments is to highlight the effectiveness of both our preconditioners within the bilevel algorithm and to compare the efficiency of P_1 and P_2 across different examples. Therefore, in the discussion of each experiment, we focus on the computation time rather than on the optimal sensor positions and the reconstructions produced, although we still visualise these for completeness.

Example 1: In this example, we use the ground truth in Figure 5.2.1 as a training model. The source positions indicated in Figure 5.2.1 are also used in this experiment. We optimise the positions of three sensors and keep the Tikhonov regularisation parameter constant. We discretise the model into a 50×50 grid (so that $M = 2601$) and use a bilevel frequency continuation strategy with three groups from 0.5 Hz to 2.5 Hz. The results of the algorithm are shown in Figure 5.2.9. The times for the whole algorithm to converge for each approach are shown in Table 5.2.11. Both preconditioning strategies are faster than using no preconditioner, reducing the time by over 40% each. In this example, the preconditioner P_1 is faster than P_2, but only by a few seconds. Since this example is on a coarse grid, M is relatively small, and so the cost of computing P_1 at the beginning of each frequency group is worth the reduction in PCG iteration produced.



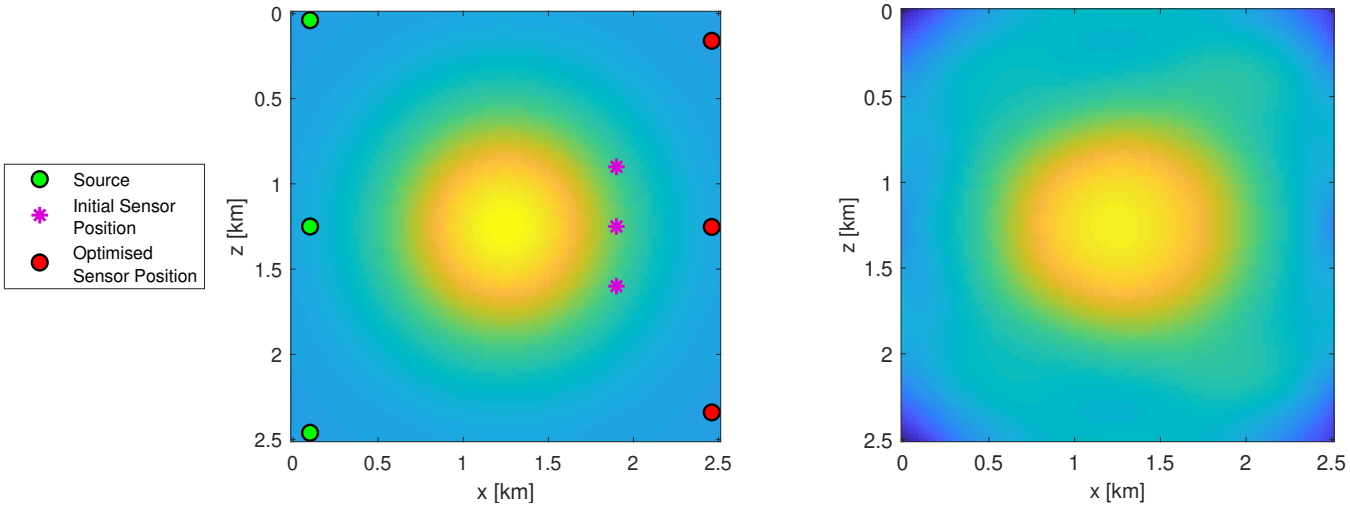
(a) Optimised Sensor Positions $\boldsymbol{p}_{\min}$ (b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min})$

Figure 5.2.9: *Optimised sensor positions and the resulting reconstruction of the training model for Example 1.*

	Preconditioning Strategy		
	None	P_1	P_2
Time (seconds)	1173	664	689

Table 5.2.11: *Time taken for Example 1 to converge for each preconditioning strategy.*

Example 2: In this example, we repeat Example 1 but on a finer grid. We use a 101×101 grid, and hence, $M = 10201$. The results of the algorithm are shown in Figure 5.2.10. The finer grid produces a better reconstruction than that seen in Example 1. The times for the whole algorithm to converge for each approach are shown in Table 5.2.12. Again, both preconditioning strategies are faster than using no preconditioner, but now there is a noticeable difference in time between the two preconditioning strategies. The P_2 preconditioner results in an algorithm that is approximately 53 minutes faster than using no preconditioner, whereas the P_1 preconditioner results in an algorithm that is approximately 32 minutes faster than using no preconditioner. The large M in this example is what makes the P_1 preconditioner slower than P_2.



(a) Optimised Sensor Positions $\boldsymbol{p}_{\min}$ (b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min})$

Figure 5.2.10: *Optimised sensor positions and the resulting reconstruction of the training model for Example 2.*

	Preconditioning Strategy		
	None	P_1	P_2
Time (seconds)	11928	10017	8730

Table 5.2.12: *Time taken for Example 2 to converge for each preconditioning strategy.*

Example 3: In this example, we use Figure 5.2.11 as the training model. This model is discretised into a 101×101 grid, and therefore $M = 10201$. We optimise five sensor positions and keep the Tikhonov regularisation parameter constant. The results of the algorithm are shown in Figure 5.2.12 and the times for the whole algorithm to converge for each approach are shown in Table 5.2.13. As in Example 2, we find that the P_2 preconditioning strategy produces the fastest algorithm. Both preconditioning strategies produce a substantial decrease in the overall time. Using P_1, the algorithm becomes approximately 3 hours 49 minutes faster than using no preconditioner, and using P_2, the algorithm becomes approximately 4 hours 45 minutes faster than using no preconditioner. The reason that the preconditioners are even more effective here than in Example 2 is the large number of gradient evaluations in this example. Example 3 took 60 upper-level iterations to reach the pre-set convergence tolerance, whereas Example 2 only involved 19 iterations. Each upper-level iteration also involves at least one line search iteration, and so the overall number of gradient evaluations grows quickly with the number of upper-level iterations. The larger the number of gradient evaluations there are, the more often the linear system must be solved and hence the more important the preconditioners become.

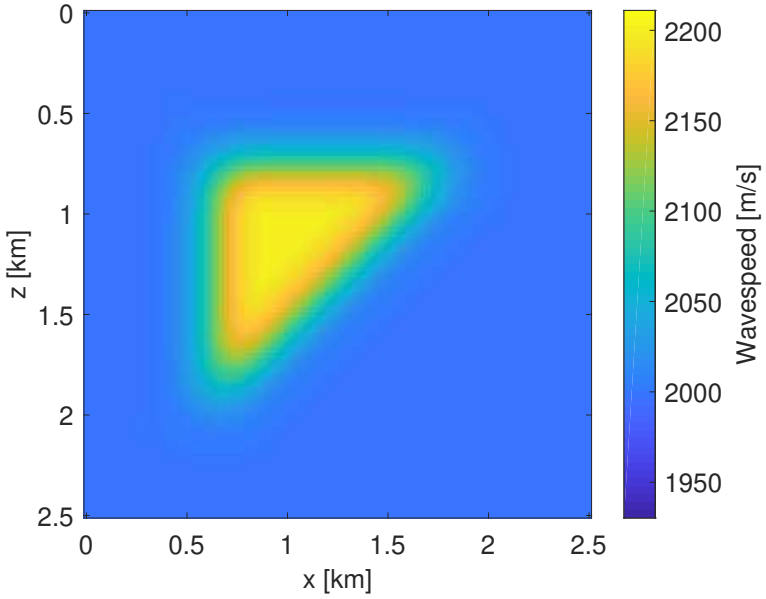
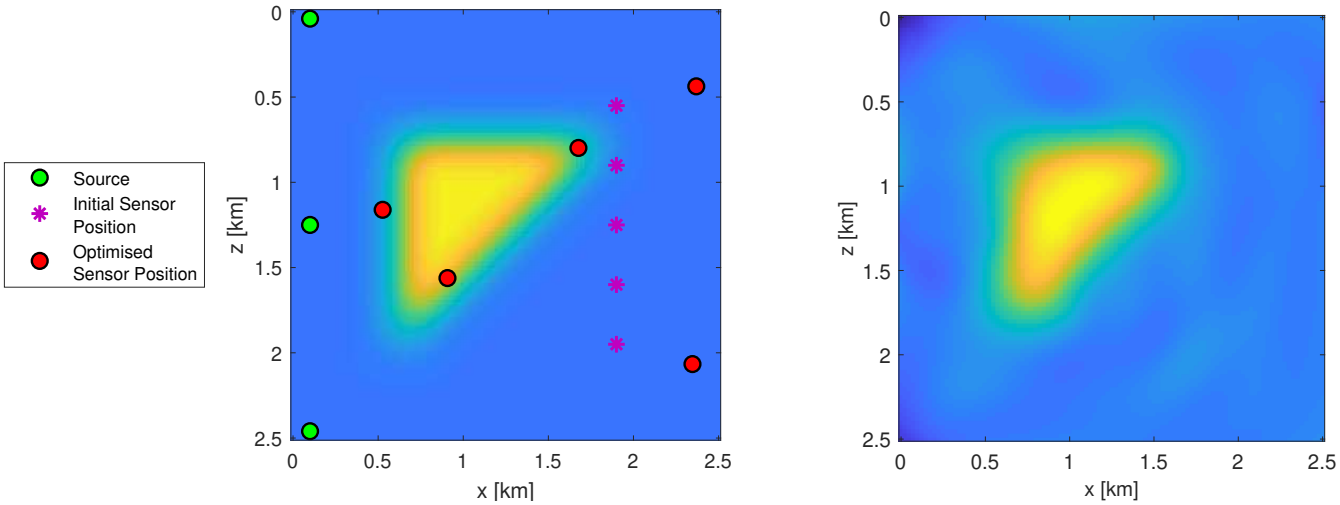


Figure 5.2.11: *Training model used for Example 3.*

	Preconditioning Strategy		
	None	P_1	P_2
Time (seconds)	43292	29548	26216

Table 5.2.13: *Time taken for Example 3 to converge for each preconditioning strategy.*



(a) Optimised Sensor Positions $\boldsymbol{p}_{\min}$ (b) Reconstruction $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min})$

Figure 5.2.12: *Optimised sensor positions and the resulting reconstruction of the training model for Example 3.*

In conclusion, the use of a preconditioner is very effective at reducing the number of iterations needed to solve the system (5.2.1) and in speeding up the overall bilevel algorithm. Based on the tables of iterations, the cost analysis and the examples presented in this section, we adapt the P_2 preconditioning strategy. In the tables of iterations taken to solve system (5.2.1), P_2 produces iterations that are always less than the non-preconditioned method. The preconditioner P_1 has the potential to result in less iterations than P_2, however P_2 appears to be more reliable. In the full sensor placement optimisation examples, P_2 has been shown to work well consistently. Although there are cases when P_1 works better, this tends to be for smaller problems, and P_2 always works just as well or better than P_1. Outside of this section, all other experiments in this thesis are performed using the preconditioner P_2.

We remark that these preconditioning strategies may be useful outside of the sensor optimisation problem and could help speed up the solution method for any system involving the Hessian, such as solving the standalone FWI problem with Newton's method as in [129].

5.3 Parallelisation and Scaling

In this section, we discuss the implementation of the full bilevel algorithm for many training models, and how this implementation can be parallelised. We include an example to demonstrate the parallel scalability of the problem.

When we are learning the optimal sensor positions/regularisation parameter from a set of training models $\mathcal{M}'$, we must perform FWI for each training model, $\boldsymbol{m}' \in \mathcal{M}'$, to produce each corresponding lower-level solution $\boldsymbol{m}^{FWI}(\boldsymbol{p}, \boldsymbol{m}')$. FWI for each training

model is completely independent of FWI for the other training models. Therefore, the lower-level can be parallelised over the training models. The upper-level cannot be parallelised as straightforwardly, as we need all training models to compute the gradient and hence to compute the update to the parameters being optimised. However, parts of the gradient computation can be parallelised. We recall that the upper-level gradient is given by (3.4.12), involving the matrix $B \in \mathbb{R}^{M \times dN_r}$ and the vector $\boldsymbol{\delta} \in \mathbb{R}^{M \times 1}$. Without going into too much detail here, the gradient computation is summarised by following steps: (i) solving (5.2.1) for $\boldsymbol{\delta}$ for each $\boldsymbol{m}'$ (as discussed in discussed in Section 5.2), (ii) computing the matrix B for each $\boldsymbol{m}'$, (iii) performing the multiplication in (3.4.12) explicitly and summing over the training models. Therefore, steps (i) and (ii) can be parallelised over the training models, and the results are then combined for step (iii). We implement this parallelisation using the `parfor` function in Matlab, which executes for-loop iterations in parallel on different 'Matlab workers'. The algorithm is then run on the University of Bath HPC cluster Balena.

Having implemented an algorithm that works across a number of workers, we now investigate the scalability of the algorithm. Scalability refers to an algorithm's performance with varying problem sizes and numbers of workers. It is important that our algorithm is scalable for parallel computing to be efficient. We examine two types of scaling here - strong scaling and weak scaling. Strong scaling refers to the algorithm's performance when the total problem size is kept fixed (i.e., the number of training models is fixed), and the number of workers is varied. Weak scaling refers to when the problem size increases at the same rate as the number of workers, keeping the amount of work per worker the same.

We test both strong and weak scaling by running the full bilevel algorithm three times, recording the time it takes to complete and reporting the average of these times here. The times that we present here are the average time taken for Approach III in Section 5.1 to converge, i.e., we are optimising the sensor positions in all frequency groups and in the final frequency group we optimise sensor positions, and the Tikhonov regularisation parameter simultaneously. We do not include the timings for the other two bilevel frequency continuation approaches, or for optimising sensor positions alone, as although the actual timings are different, the behaviour and conclusions are the same for all approaches.

5.3.1 Strong Scaling

For the strong scaling experiment, we measure the time taken for a problem with 16 training models to run for various numbers of Matlab workers. These training models are displayed in Figure 5.3.1 and involve a Gaussian bump with maximum wavespeed of 2100 m/s, centred at various positions along a horizontal line. Each of these models are discretised into 50×50 grids. Three sources are used for acquisition and the positions of three sensors and the Tikhonov regularisation parameter are optimised.

The average times for the bilevel algorithm to run are given in Table 5.3.1. Times are given to the nearest second. As we add more Matlab workers, the work done per each worker is reduced and hence the computational runtime is decreased.

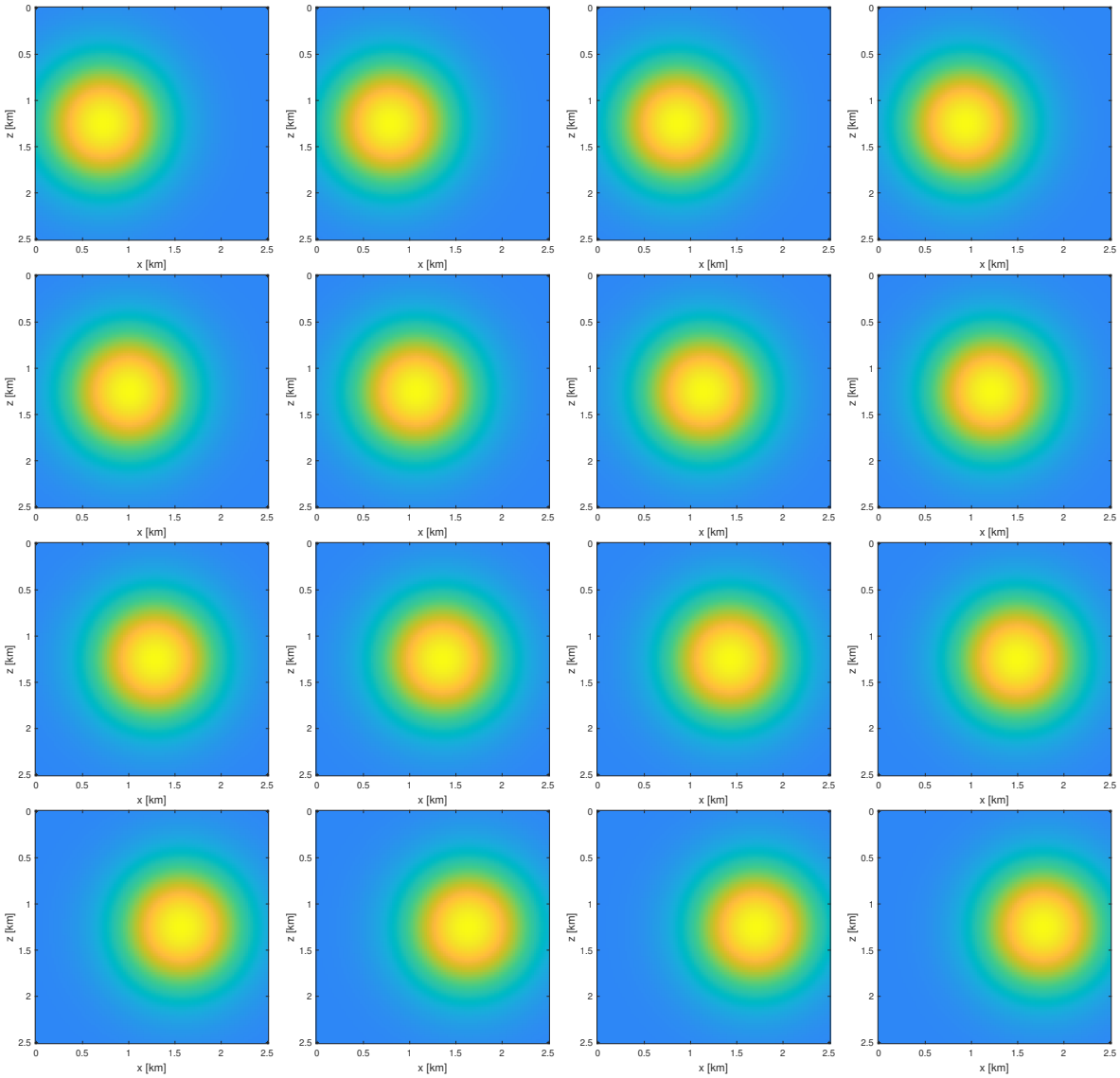


Figure 5.3.1: *Set of training models $\mathcal{M}'$ used for strong scaling experiments.*

	Number of Matlab workers n				
	1	**2**	**4**	**8**	**16**
Runtime (s)	61208	33759	18973	7151	4254

Table 5.3.1: *Strong scaling runtime results.*

Extrapolating the results in Table 5.3.1, we observe the following relationship between the runtime and the number of workers,

$$\text{Runtime} \propto n^{-0.99}.$$

If the algorithm could be completely parallelised, we would expect the runtime to be exactly inversely proportional to number of workers, but in practice this is rarely

achieved due to parallel overhead and serial parts of the algorithm. In our case, the runtime scales very well with the number of workers, despite the fact that only certain sections of the code are parallelised over the training models.

To examine the results further, we introduce the concept of *speedup*. Simply, speedup measures how much faster the algorithm becomes when run on more workers. The formula for speedup is

$$\text{Speedup} = \frac{\tau_1}{\tau_n},$$

where τ_1 is the computational time for running the algorithm on one worker, and τ_n is the computational time for running the algorithm on n workers. Linear speedup, with speedup equal to the number of workers exactly, is ideal as because that would mean that every worker would be contributing 100% of its computational power. We include the values of speedup for our implementation in Table 5.3.2, where values are rounded to two decimal places.

	Number of Matlab workers n				
	1	**2**	**4**	**8**	**16**
Speedup	1	1.81	3.22	8.56	14.39

Table 5.3.2: *Strong scaling speedup.*

Taking a least squares fit of the data in Table 5.3.2, we find that the gradient is 0.99, suggesting that the speedup scales like $n^{0.99}$. This relationship implies that the speedup that we are observing in practice is close to the ideal. This may be overly optimistic as the value for 8 workers in this example is better than expected. Excluding this value from the least squares fit gives the relationship that speedup scales like $n^{0.96}$. We note that the speedup observed for 8 workers is called superlinear speedup and is related to the fact that there is more cache memory available as the number of workers is increased, enabling the faster access of data [156].

The final measurement of scalability that we examine is the *efficiency* of parallel implementation. The strong scaling efficiency is defined as the ratio of the ideal runtime to the measured runtime. The ideal time for the algorithm to run on several workers is the time you expect the algorithm should take based on the measured time at the smallest worker count. Therefore, the formula for strong scaling efficiency can be written as

$$\text{Strong Scaling Efficiency} = \frac{\tau_1}{n\tau_n},$$

and is often expressed as a percentage. We find that the efficiency at 16 workers is 89.93%. Note that anything approaching 100% efficiency is rarely achieved in practice.

5.3.2 Weak Scaling

We test weak scaling with the following example. Again, three sources are used for acquisition and the positions of three sensors and the Tikhonov regularisation parameter

are optimised. The training model used is shown in Figure 5.3.2. When running the algorithm with larger numbers of training models, we use copies of the model in Figure 5.3.2 to create a set. In this way we can be certain that we are solving the same problem as the number of training models is increased. (Although in practice the training set is made up of different models, if we were to use various different training models here, the objective function, optimal parameters and the difficulty of finding those parameters would change and we would not be able to adequately measure how the algorithm scales.)

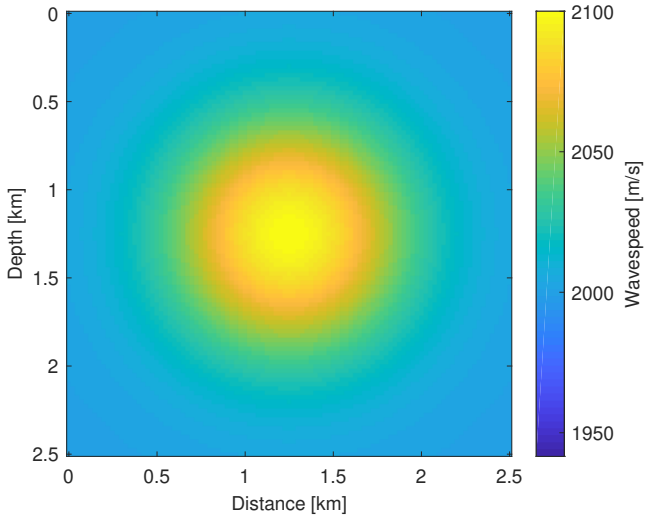


Figure 5.3.2: *Training model $\boldsymbol{m}'$ used for scaling experiments*

As we vary the number of training models, we also vary the number of workers, so that the problem size per worker remains constant. The average runtimes (to the nearest second) are given in Table 5.3.3.

Number of training models	**Number of Matlab workers** n				
	1	**2**	**4**	**8**	**16**
1	2565				
2	5103	2601			
4	9818	5057	2717		
8	19709	9886	5440	2849	
16	36666	19819	10775	5551	2921

Table 5.3.3: *Weak scaling runtime results.*

If the problem were to scale perfectly, the runtimes along the diagonal would be constant. This is not the case here, and we see some growth in time as the number of workers and training models is increased. However, the growth is slow (approximately growing as $n^{0.05}$).

To measure weak scaling, we define a quantity called *scaled speedup*, given by

$$\text{Scaled Speedup} = \frac{\tau_1(n)}{\tau_n(n)},$$

where $\tau_1(n)$ is the computational time that a problem of size n takes running on one worker, and $\tau_n(n)$ is the computational time that a problem of size n takes running on n workers. This is called scaled speedup as it is calculated based on the amount of work done for a scaled problem size (in contrast to the previous definition of speedup which focuses on a fixed problem size). The scaled speedup is plotted in Figure 5.3.3. The approximate relationship that we observe is

$$\text{Scaled Speedup} \propto n^{0.91}.$$

The scaled speedup grows as the number of workers/problem size is increased. It doesn't grow exactly proportional to the number of workers however, due to the increased parallel overhead as the number of workers is increased.

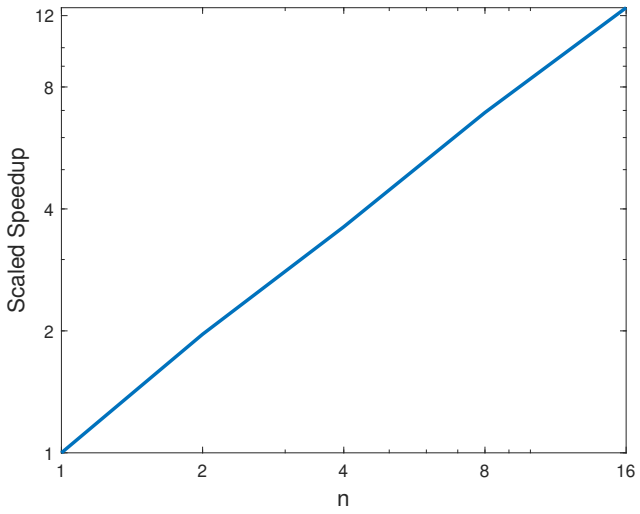


Figure 5.3.3: *Log-log plot of scaled speedup.*

The weak scaling efficiency is defined here as the ratio between the time to complete a problem with one training model on one worker to the time taken to complete a problem with n training models on n workers, i.e.,

$$\text{Weak Scaling Efficiency} = \frac{\tau_1(1)}{\tau_n(n)}.$$

The plot of weak scaling efficiency versus n for this problem is shown in Figure 5.3.4. We can see that there is a trend of decreasing efficiency as the problem size/number of works is increased. The ideal efficiency would stay at 100% since the problem size per worker remains constant, but in reality the efficiency decreases with an increasing number of workers. This is related to the increased amount of communication and data transfer between the workers. For our problem, even though the efficiency decreases, it remains high at approximately 87.81% for $n = 16$.

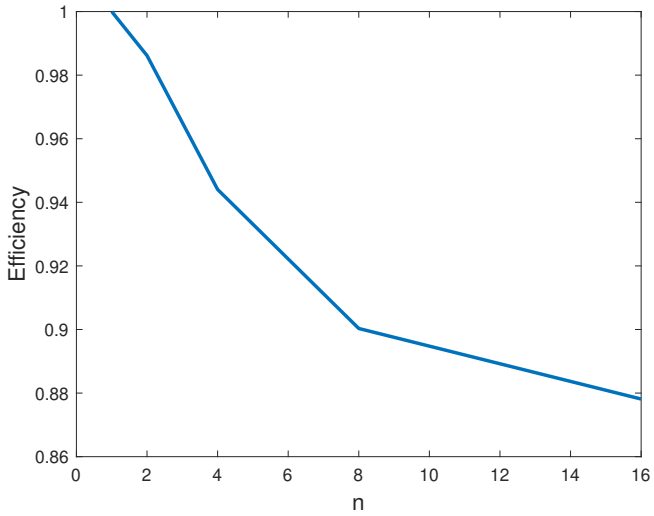


Figure 5.3.4: *Weak scaling efficiency.*

5.3.3 Breakdown of Computational Cost

We have shown that the parallel implementation of our whole bilevel algorithm scales very well, for both strong and weak scaling tests. Here we investigate why the whole bilevel algorithm scales so well, despite the fact that only certain sections of the code are parallelised over the training models. For various experiments in this thesis, we used the Matlab Profiler to track execution time for each part of the algorithm. We have observed that between approximately 75 and 94% of the runtime of the full bilevel algorithm is spent in performing FWI, which is completely parallelised over the training models. The majority of the rest of the time, often somewhere between 6 and 25% of the runtime, is spent computing the upper-level gradient, most of which is also parallelised. Therefore, we observe that, in general, up to 99% of the runtime of the bilevel algorithm has been parallelised, explaining why it scales so well.

We next discuss why the FWI and gradient computation dominate the runtime. Recall that the number of PDE solves in the bilevel algorithm is given by (5.2.11). All of these PDE solves occur either in the FWI or upper-level gradient computation. The breakdown of where the PDE solves occur is as follows:

(1) **FWI:** $N_{m'} N_u^{ls} N_s N_\omega \, (2N_l)$ PDE solves

(2) **Upper-level gradient:** $N_{m'} N_u^{ls} N_s N_\omega \left(2N_i^{P_2} + dN_r\right)$ PDE solves

(where the notation used is defined in Theorem 3.4.10). We have observed in our experiments that the FWI computation makes up a larger proportion of the total runtime than the upper-level gradient computation. Therefore, cancelling the common terms in (1) and (2), we conclude that, in our computations,

$$N_l > N_i^{P_2} + \frac{dN_r}{2}, \tag{5.3.1}$$

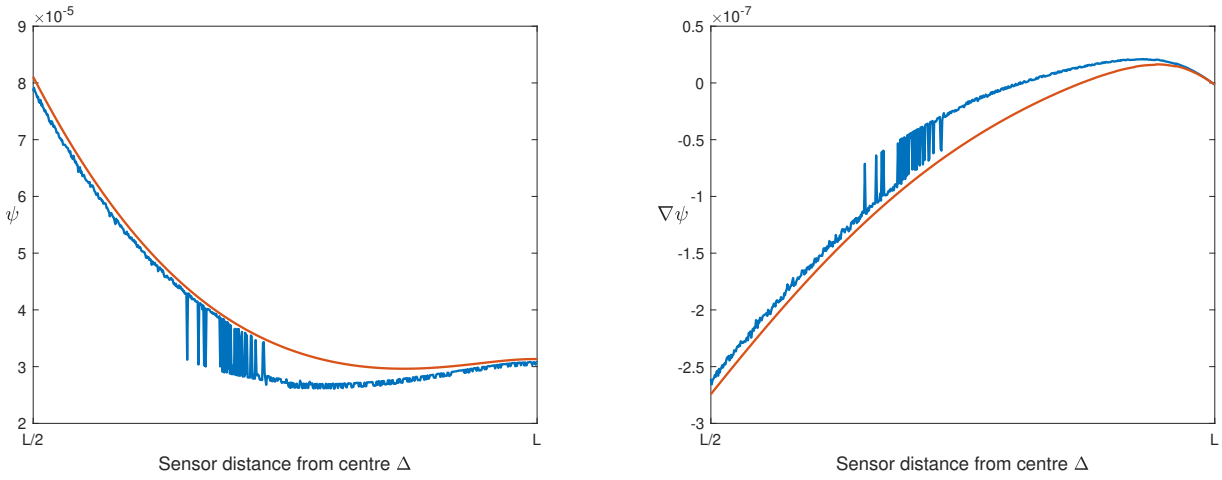
where the left-hand side is the number of FWI iterations, N_l, and the right-hand side involves the number of preconditioned conjugate gradient iterations, $N_i^{P_2}$, and the number of parameters being optimised, dN_r. We expect the inequality (5.3.1) is due to two reasons. The first reason is that we have developed techniques to reduce the cost of the upper-level gradient computation (i.e., the right-hand side of (5.3.1)). The number of PDE solves and hence the runtime of the upper-level gradient computation was reduced significantly in Section 5.2.2 through the use of a preconditioner, so the factor $N_i^{P_2}$ should not be too large. However, we note from Section 5.2.2 that this factor varies with the parameters of the problem (such as the regularisation parameters and the frequency), and so for problems not considered in this thesis, such as higher frequency problems, this factor may contribute more to the overall % of the run time that the upper-level gradient takes. For problems that are symmetric (in the sense described in Chapter 4), we showed how the factor dN_r could be reduced in Chapter 4, however for non-symmetric problems this cannot be reduced and depends directly on the number of sensors. When there are more sensors, the % of the run time that the upper-level gradient takes increases.

The second reason that FWI dominates our computational runtime is that we need to solve the FWI problem to a high accuracy, which is what makes N_l is large (this idea of needing to solve the lower-level very accurately is also noted in [162]). This is related to the single-level reduction in Section 3.4.1. In this reduction, we define the lower-level solution $\boldsymbol{m}^{FWI}$ with the following necessary condition,

$$\nabla\phi(\boldsymbol{m}^{FWI}(\boldsymbol{p},\boldsymbol{m}'),\boldsymbol{p},\boldsymbol{m}') = \boldsymbol{0}, \tag{5.3.2}$$

which allows us to derive a formula for the upper-level gradient. Therefore, the formula for our upper-level gradient is only correct when (5.3.2) holds, and, to compute the correct upper-level gradient, we require the FWI algorithm to run to the point where the gradient of the FWI objective function is zero. In practice, we cannot solve to exactly zero, but we solve to a very small number, the exact value of which is problem dependant. Solving to such a high accuracy often requires several hundred FWI iterations (i.e., N_l can be several hundred). In Section 2.6, we showed that it is possible to get good quality FWI reconstructions solving to a much larger tolerance (a tolerance of $||\nabla\phi||_2 \leq 10^{-6}$ to be exact for those examples). However, in practice a tolerance this large results in an inaccurate upper-level, and hence causes the bilevel algorithm to fail. We include an example here to demonstrate this point.

We draw ψ versus sensor position for the example shown in Figure 5.1.1. The 0.5 Hz ψ plot shown in Figure 5.1.2 has been drawn with an FWI tolerance of $||\nabla\phi||_2 \leq 10^{-10}$. We redraw this with an FWI tolerance $||\nabla\phi||_2 \leq 10^{-6}$. We also compute and plot the corresponding values of $\nabla\psi$. These plots are shown in Figure 5.3.5, where we have focused on a segment of the plot to highlight the difference in the results for the different tolerances. We see that when FWI is solved to the 10^{-10} tolerance (in red), both ψ and $\nabla\psi$ are smooth. When solved to the 10^{-6} tolerance (in blue), ψ and $\nabla\psi$ are extremely rough. Therefore, for the bilevel problem to work, we need FWI to be solved much more accurately than it is generally solved in practice for the standalone FWI problem. The convergence tolerance that is sufficient for a good FWI reconstruction, is generally not sufficient for the bilevel problem.



(a) ψ (b) $\nabla\psi$

Figure 5.3.5: *Plots of ψ and $\nabla\psi$ computed using different FWI tolerances. The tolerance $||\nabla\phi||_2 \leq 10^{-6}$ is plotted in blue and $||\nabla\phi||_2 \leq 10^{-10}$ is plotted in red.*

We noted in Section 2.6 when discussing plots of $||\nabla\phi||_2$ that the values of $\nabla\phi$ continue to decrease even when the decrease in ϕ has stalled. Therefore, to reach the required convergence tolerance for the bilevel problem, many more iterations are needed than the standalone problem. In the specific example in Figure 5.3.5, it takes between approximately 90 to 150 iterations to solve FWI to the tolerance 10^{-6}, and between approximately 250 to 315 iterations to solve FWI to the tolerance 10^{-10}. This example demonstrates the large N_l and hence explains the large proportion of runtime that is needed for FWI in the bilevel algorithm.

5.4 Algorithms

This section provides details on the bilevel sensor placement optimisation algorithms and further implementation details.

Notation: We first present Table 5.4.1, a table of notation to be used in this section. We include a summary of important points and new notation here. In the algorithms presented here, we optimise both the sensor positions $\boldsymbol{p}$ and Tikhonov regularisation parameter α. As both $\boldsymbol{p}$ and α are optimised together, we introduce the new notation $\boldsymbol{q}$ to denote the optimisation variables, which is the concatenated vector $[\boldsymbol{p}; \alpha]$. We write $\boldsymbol{q}$ when optimising these parameters together, but we continue to write $\boldsymbol{p}$ and α separately when they are needed individually.

Another important notation point is that we introduce the set of initial models $\mathcal{M}_0$ and set of FWI solutions $\mathcal{M}^{FWI}$, such that each element of the training model set $\mathcal{M}'$ has a corresponding member of the sets $\mathcal{M}_0$ and $\mathcal{M}^{FWI}$, i.e., the training model $\boldsymbol{m}'_i$ has a corresponding initial guess $(\boldsymbol{m}_0)_i$ and FWI solution $\boldsymbol{m}_i^{FWI}$, for $i = 1, \ldots, N_{m'}$. We write $\mathcal{M}^{FWI}(\boldsymbol{q})$ to mean the FWI solutions computed with the parameters $\boldsymbol{q}$.

For ease of notation, the regularisation terms of the Hessian (i.e., the terms that form preconditioner P_2 in Section 5.2.2) is denoted by $\mathbf{P}$ here. Also, we write $\mathbf{u}(\boldsymbol{m}, \mathcal{S}, g)$ to denote the wavefields at all sources $s \in \mathcal{S}$ and for all frequencies in the group g (i.e., this notation represents several wavefields), and $\mathbf{u}(\boldsymbol{m}, s, \omega)$ to mean the wavefield at a specific source and frequency (i.e., just one wavefield).

In some algorithms the list of all inputs in practice is very long, so we only write them if they are necessary to the understanding of that particular algorithm.

Symbol	Meaning
$\boldsymbol{p}$, $\boldsymbol{p}_0$, $\boldsymbol{p}_k$, $\boldsymbol{p}_{\min}$	Sensor coordinate vectors; general, initial, kth iterate, optimal
α, α_0, α_k, $\alpha_{\min}$	Regularisation parameter for Tikhonov term; general, initial, kth iterate, optimal
$\boldsymbol{q}$, $\boldsymbol{q}_0$, $\boldsymbol{q}_k$, $\boldsymbol{q}_{\min}$	Optimisation variables $[\boldsymbol{p}; \alpha]$; general, initial, kth iterate, optimal
tol_1, tol_2, tol_3	Tolerances for stopping conditions; lower-level, upper-level, PCG
$\boldsymbol{m}$, $\boldsymbol{m}_0$, $\boldsymbol{m}_k$, $\boldsymbol{m}^{FWI}$, $\boldsymbol{m}'$	Models; general, initial, kth iterate, FWI solution, training
$\mathcal{M}'$, $\mathcal{M}_0$, $\mathcal{M}^{FWI}$, $\mathcal{M}^{FWI}(\boldsymbol{q}_k)$	Set of models $\{\boldsymbol{m}_i\}_{i=1,\ldots,N_{m'}}$; training, initial, FWI solution, FWI solution at parameters $\boldsymbol{q}_k$
$k_{\max}$, $k'_{\max}$	Maximum iterations for stopping conditions; lower-level, upper-level
μ	Regularisation parameter for convex term
$\mathcal{W}$	Set of frequencies
ω	Individual frequency from set $\mathcal{W}$ such that $\omega_1 < \omega_2 < \ldots < \omega_{N\omega}$ is in increasing order
g	Frequency group of size N_g such that $g_1 < g_2 < \ldots < g_{N_f}$ is in increasing order
$\mathcal{S}$	Set of sources
s	Individual source in set $\mathcal{S}$
$\mathbf{u}$	Forward wavefield
$\boldsymbol{\lambda}$	Adjoint wavefield
$\boldsymbol{\varepsilon}$	Residual
$\phi(\boldsymbol{m})$, $\nabla\phi(\boldsymbol{m})$	Lower-level objective function and gradient (note $\nabla\phi(\boldsymbol{m})$ means $\nabla_{\boldsymbol{m}}\phi(\boldsymbol{m})$)
$\psi(\boldsymbol{q})$, $\nabla\psi(\boldsymbol{q})$	Upper-level objective function and gradient (note $\nabla\psi(\boldsymbol{q})$ means $\nabla_{\boldsymbol{q}}\psi(\boldsymbol{q}) = [\nabla_{\boldsymbol{p}}\psi(\boldsymbol{p}, \alpha); \nabla_\alpha\psi(\boldsymbol{p}, \alpha)]$)
$\mathbf{d}$	'Observed' data
$\mathbf{P}$	Preconditioner for upper-level gradient computation

Table 5.4.1: *Notation used in the algorithms in §5.4.*

Algorithms: The algorithms that form the full bilevel learning algorithm are as follows. Algorithm 5.4.1 is our novel bilevel frequency continuation algorithm, motivated and described in detail in Section 5.1. Algorithm 5.4.1 organises frequencies into groups of increasing magnitude, looping over the frequency groups and solving the bilevel optimisation problem (Algorithm 5.4.2) on each group. The solution to the upper and lower-levels (i.e., the optimal sensor positions/Tikhonov parameter and optimal models) for the first frequency group are used as the starting guesses for the next, higher, frequency group. We included a simple version of the bilevel frequency continuation algorithm in Algorithm 5.1.1 to demonstrate these concepts, while the version presented here, Algorithm 5.4.1, contains more detail. For instance, Algorithm 5.4.1 is written for a set of training models instead of just one. In addition, the first step of Algorithm 5.4.1 is now the computation of the preconditioner. As described in Section 5.2.2, our choice of preconditioner only needs to be computed once at the beginning of the whole algorithm using the regularisation parameters. The Cholesky factorisation of the preconditioner is computed and only the Cholesky factor needs to be stored and passed to the following algorithms. We note that in Algorithm 5.4.1 both $\boldsymbol{p}$ and α are optimised together in all frequency groups, i.e., we are using Approach I from Section 5.1. See Remark 5.4.2 for an algorithm that considers all approaches from Section 5.1 for incorporating the Tikhonov parameter into the bilevel frequency continuation.

Algorithm 5.4.2 is the bilevel optimisation algorithm, which we solve using a gradient-based optimisation method with line search. When we refer to 'Descent Direction', any gradient-based optimisation method can be used for computing the descent direction. We provide an overview of different choices of optimisation methods in Appendix F. In our implementation, we use L-BFGSb, which is the bounded version of L-BFGS. We require the bounded version to ensure that the sensors stay within the domain we are considering, or within some realistic range. Our implementation is based on [39] and uses sections of [80]. The step 'Line Search' in Algorithm 5.4.2 can represent any line search algorithm that computes the step-size. In our implementation we use the strong Wolfe conditions (also discussed in Appendix F). Since we use a gradient-based optimisation method, at each iteration of Algorithm 5.4.2 (i.e., for each updated $\boldsymbol{q}$) we are required to compute the gradient of the upper-level objective function ψ, which we do by calling Algorithm 5.4.3.

There are two main parts to Algorithm 5.4.3, the full lower-level/FWI problem solution and the computation of the upper-level gradient. These two steps are performed for every training model. As discussed in Section 5.3, the lower-level and most of the upper-level gradient computation can be parallelised over the training models, and this parallelisation is indicated by the **parfor** in Algorithm 5.4.3. Before the lower-level step, the data that is input into the FWI algorithm is computed for each training model (according to (2.2.14) and (3.3.6)). We don't include the finer details of this data computation in the algorithm, but we note that, since the wavefield for each training model, $\mathbf{u}(\boldsymbol{m}')$, stays the same throughout the optimisation, this only needs to be computed once (for all frequencies and sources) earlier in the process and passed as an input into Algorithm 5.4.3. The data computation then just involves applying the restriction operator to the relevant wavefields (which is cheap as no PDE solves are required). The lower-level step is then completed with Algorithm 5.4.4. More

explanation of the FWI implementation is contained in Section 2.5.2 and we just mention here how the FWI algorithm ties into the bilevel algorithm. Algorithm 5.4.4 outputs the optimal model $\boldsymbol{m}^{FWI}$ (which is saved in the set $\mathcal{M}^{FWI}$) along with the residual $\boldsymbol{\varepsilon}$, and the solution to two PDEs (for each source and frequency) - the forward wavefield $\mathbf{u}$ and the adjoint wavefield $\boldsymbol{\lambda}$, all evaluated at the optimal model $\boldsymbol{m}^{FWI}$. The outputs of the lower-level problem are then used as inputs into Algorithm 5.4.6, which is used to compute the upper-level objective function and gradient for each training model. These values are then used in Algorithm 5.4.3 to compute the full upper-level objective function and gradient for all models.

Remark 5.4.1. *We note here that the lower-level (FWI problem) in the bilevel problem must be solved very accurately due to the assumption we made for the single-level reduction (3.4.1) that the lower-level FWI problem can be replaced by its necessary condition. To ensure this assumption holds, we require* tol_1 *in Algorithm 5.4.4 to be as close to zero as possible. There is a more detailed discussion of this point in Section 5.3.3.*

Within Algorithm 5.4.6, we solve the linear system (3.4.20) with the Preconditioned Conjugate Gradient (PCG) method (Algorithm 5.4.7). This is a standard PCG algorithm, where the matrix-vector products (i.e., Hessian-vector products where the Hessian is evaluated at the FWI solution) are computed using Algorithm 5.4.8 (based on the theory in Section 2.4.2).

The relationship between all the algorithms in this section is illustrated in Figure 5.4.2 which shows how each algorithm depends on the others. Figure 5.4.1 shows the overall behaviour of the bilevel algorithm at a high-level. This figure demonstrates how the upper- and lower-level problems interact in the bilevel problem. The initial set of sensor positions and regularisation parameter is input into the lower-level problem, and the FWI problem is solved for this specific set of parameters, for each training model. The set of FWI solution models is then used in the upper-level algorithm to compute the sensor placement objective function (Definition 3.3.2) and its gradient. The gradient-based optimisation method computes a new $\boldsymbol{p}$ and α, such that the upper-level objective function is reduced. These updated parameters are input back into the lower-level problem. This cycle continues until some pre-set convergence criteria is met, and the optimal set of sensor positions is output. This cycle is all contained within Algorithm 5.4.1 and is repeated progressing through frequencies. Figure 5.4.1 highlights that on every upper-level iteration, a full FWI problem needs to be solved for every training model. By Remark 5.4.1, each FWI problem needs to be solved very accurately. Therefore the implementation of the lower-level problem contributes a lot to the overall cost/time of the bilevel problems, as observed in Section 5.3.

Algorithm 5.4.1 Bilevel Frequency Continuation

Inputs: $\boldsymbol{p}_0$, α_0, $\mathcal{M}_0$, $\mathcal{M}'$, $\{\omega_1 < \omega_2 < ... < \omega_{N\omega}\} \in \mathcal{W}$, μ
Compute preconditioner $\mathbf{P}$ with α_0 and μ
$L \leftarrow$ Cholesky($\mathbf{P}$)
Group frequencies into N_f groups $\{g_1, g_2, \ldots, g_{N_f}\}$
$\boldsymbol{q}_0 \leftarrow [\boldsymbol{p}_0; \alpha_0]$
for $k = 1$ **to** N_f **do**
    $[\boldsymbol{q}_{\min}, \mathcal{M}^{FWI}] \leftarrow$ `Algorithm 5.4.2`$(g_k, \boldsymbol{q}_0, \mathcal{M}_0, \mathcal{M}', L)$
    $\boldsymbol{q}_0 \leftarrow \boldsymbol{q}_{\min}$
    $\mathcal{M}_0 \leftarrow \mathcal{M}^{FWI}$
end for
Output: $\boldsymbol{p}_{\min}$, $\alpha_{\min}$

Algorithm 5.4.2 Bilevel Optimisation Algorithm

Inputs: g, $\boldsymbol{q}_0$, $\mathcal{M}_0$, $\mathcal{M}'$, L, $\mathcal{S}$, $k'_{\max}$, tol_1, tol_2
Compute $\nabla\psi(\boldsymbol{q}_0)$ with `Algorithm 5.4.3`$(\boldsymbol{q}_0, \mathcal{M}_0, tol_1, \mathcal{M}', g, \mathcal{S}, L)$
$k = 0$
while $||\nabla\psi(\boldsymbol{q}_k)|| > tol_2$ and $k < k'_{\max}$ **do**
    $\mathbf{d}'_k \leftarrow$ Descent Direction
    $\beta'_k \leftarrow$ Line Search
    $\boldsymbol{q}_{k+1} \leftarrow \boldsymbol{q}_k + \beta'_k \mathbf{d}'_k$
    $[\nabla\psi(\boldsymbol{q}_{k+1}), \mathcal{M}^{FWI}(\boldsymbol{q}_{k+1})] \leftarrow$ `Algorithm 5.4.3`$(\boldsymbol{q}_k, \mathcal{M}_0, tol_1, \mathcal{M}', g, \mathcal{S}, L)$
    $k = k + 1$
end while
Outputs: $\boldsymbol{q}_{\min}$, $\mathcal{M}^{FWI}(\boldsymbol{q}_{\min})$

Algorithm 5.4.3 Computation of ψ, $\nabla\psi$ and $\mathcal{M}^{FWI}$

Inputs: $\boldsymbol{q}$, $\mathcal{M}_0$, tol_1, $\mathcal{M}'$, g, $\mathcal{S}$, L
$\boldsymbol{p} = \boldsymbol{q}$(1:end-1)
$\alpha = \boldsymbol{q}$(end)
Initialise ψ=0, $\nabla\psi$=0
parfor $\boldsymbol{m}' \in \mathcal{M}'$, $\boldsymbol{m}_0 \in \mathcal{M}_0$
    Compute $\mathbf{d}(\boldsymbol{m}')$ by (3.3.6)
    $[\boldsymbol{m}^{FWI}, \boldsymbol{\varepsilon}(\boldsymbol{m}^{FWI}), \mathbf{u}(\boldsymbol{m}^{FWI}), \boldsymbol{\lambda}(\boldsymbol{m}^{FWI})] \leftarrow$ `Algorithm 5.4.4`$(\boldsymbol{m}_0, tol_1, \mathbf{d}(\boldsymbol{m}'), \mathcal{S},$
    $g, \boldsymbol{p}, \alpha)$ *(Lower-Level)*
    $[\psi', \nabla\psi'] \leftarrow$ `Algorithm 5.4.6`$(\boldsymbol{p}, \alpha, \boldsymbol{m}^{FWI}, \boldsymbol{m}', \boldsymbol{\varepsilon}(\boldsymbol{m}^{FWI}), \mathbf{u}(\boldsymbol{m}^{FWI}), \boldsymbol{\lambda}(\boldsymbol{m}^{FWI}), L)$
    Update $\psi = \psi + \psi'$
    Update $\nabla\psi = \nabla\psi + \nabla\psi'$
    Save $\boldsymbol{m}^{FWI}$ in $\mathcal{M}^{FWI}$
end parfor
Scale $\psi = \frac{1}{N_{m'}}\psi$
Scale $\nabla\psi = \frac{1}{N_{m'}}\nabla\psi$
Outputs: ψ, $\nabla\psi$, $\mathcal{M}^{FWI}$

Algorithm 5.4.4 Lower-Level (FWI)

Inputs: $\boldsymbol{m}_0$, tol_1, $\mathbf{d}$, $\mathcal{S}$, g, $\boldsymbol{p}$, α, μ, $k_{\max}$
Compute $\nabla\phi(\boldsymbol{m}_0)$ with `Algorithm 5.4.5`$(\boldsymbol{m}_0, \mathbf{d}, \mathcal{S}, g, \boldsymbol{p}, \mu, \alpha)$
$k = 0$
while $||\nabla\phi|| > tol_1$ and $k < k_{\max}$ **do**
    $\mathbf{d}_k \leftarrow$ Descent Direction
    $\beta_k \leftarrow$ Line Search
    $\boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \beta_k \mathbf{d}_k$
    $[\nabla\phi(\boldsymbol{m}_{k+1}), \boldsymbol{\varepsilon}(\boldsymbol{m}_{k+1}), \mathbf{u}(\boldsymbol{m}_{k+1}), \boldsymbol{\lambda}(\boldsymbol{m}_{k+1})] \leftarrow$ `Algorithm 5.4.5`$(\boldsymbol{m}_{k+1}, \mathbf{d}, \mathcal{S}, g, \boldsymbol{p}, \mu, \alpha)$
    $k = k + 1$
end while
Outputs: $\boldsymbol{m}^{FWI}$, $\boldsymbol{\varepsilon}(\boldsymbol{m}^{FWI}, \mathcal{S}, g)$, $\mathbf{u}(\boldsymbol{m}^{FWI}, \mathcal{S}, g)$, $\boldsymbol{\lambda}(\boldsymbol{m}^{FWI}, \mathcal{S}, g)$

Algorithm 5.4.5 FWI Objective function and Gradient

Inputs: $\boldsymbol{m}$, $\mathbf{d}$, $\mathcal{S}$, g, $\boldsymbol{p}$, μ, α
Initialise $\phi = \frac{1}{2}\alpha||D\boldsymbol{m}||_2^2 + \frac{1}{2}\mu||\boldsymbol{m}||_2^2$, $\nabla\phi = \alpha D^T D\boldsymbol{m} + \mu\boldsymbol{m}$
Compute restriction operator $R(\boldsymbol{p})$
for $s \in \mathcal{S}$ **do**
    for $k \in \{1, .., N_g\}$ **do**
        Assemble source vector $\mathbf{f}(s, \omega_k)$
        Assemble matrix $A(\boldsymbol{m}, \omega_k)$
        Compute $\mathbf{u}(\boldsymbol{m}, s, \omega_k)$ from (2.2.14)
        Compute modelled data $\mathbf{d}^{mod}(\boldsymbol{m}, \boldsymbol{p}, s, \omega_k) = R(\boldsymbol{p})\mathbf{u}(\boldsymbol{m}, s, \omega_k)$
        Evaluate misfit $\boldsymbol{\varepsilon}(\boldsymbol{m}, \boldsymbol{p}, s, \omega_k) = \mathbf{d} - \mathbf{d}^{mod}(\boldsymbol{m}, \boldsymbol{p}, s, \omega_k)$
        Compute adjoint wavefield $\boldsymbol{\lambda}$ form (2.3.12)
        Update $\phi = \phi + \frac{1}{2}||\boldsymbol{\varepsilon}||_2^2$
        for $j \in \{1, ..., M\}$ **do**
            Update $\nabla\phi_j = \nabla\phi_j + \left(\frac{\partial A}{\partial m_j}\mathbf{u}\right)^* \boldsymbol{\lambda}$
        end for
    end for
end for
Outputs: ϕ, $\nabla\phi$, $\mathbf{u}$, $\boldsymbol{\varepsilon}$, $\boldsymbol{\lambda}$

Algorithm 5.4.6 Upper-Level Objective Function and Gradient for each Training Model

Inputs: $\boldsymbol{p}$, α, $\boldsymbol{m}^{FWI}$, $\boldsymbol{m}'$, $\boldsymbol{\varepsilon}(\boldsymbol{m}^{FWI}, \boldsymbol{p}, \mathcal{S}, g)$, $\mathbf{u}(\boldsymbol{m}^{FWI}, \boldsymbol{p}, \mathcal{S}, g)$, $\boldsymbol{\lambda}(\boldsymbol{m}^{FWI}, \boldsymbol{p}, \mathcal{S}, g)$, μ, L
Compute restriction operator $R(\boldsymbol{p})$
Compute derivative of restriction operator $\frac{dR(\boldsymbol{p})}{dp_n}$ for all n
$\boldsymbol{\delta} \leftarrow$ `Algorithm 5.4.7`$(\boldsymbol{m}^{FWI}, \boldsymbol{m}', \mathbf{u}(\boldsymbol{m}^{FWI}), \boldsymbol{\lambda}(\boldsymbol{m}^{FWI}), \mathcal{S}, g, \mu, \alpha, L, R(\boldsymbol{p}))$
Initialise $B = 0$
for $s \in \mathcal{S}$
    for $k \in \{1, \ldots, N_g\}$
        Assemble matrix $A(\boldsymbol{m}^{FWI}, \omega_k)$
        for $n = 1, .., dN_r$
            Compute $\boldsymbol{\gamma}_n(\boldsymbol{m}^{FWI}, s, \omega_k)$ via (3.4.22)
            for $j \in 1, \ldots, M$
                $B_{j,n} = B_{j,n} + \left(\frac{\partial A}{\partial m_j}\mathbf{u}(\boldsymbol{m}^{FWI}, s, \omega_k)\right)^* \boldsymbol{\gamma}_n$
            end for
        end for
    end for
end for
Compute $\nabla\psi_{\boldsymbol{p}} \in \mathbb{R}^{dN_r}$ by (3.4.19) with $N_{m'} = 1$
Compute $\nabla\psi_\alpha \in \mathbb{R}^1$ by (3.5.10) with $N_{m'} = 1$
$\nabla\psi = [\nabla\psi_{\boldsymbol{p}}; \nabla\psi_\alpha]$
Compute ψ by (3.3.1) with $N_{m'} = 1$
Outputs: ψ, $\nabla\psi$

Algorithm 5.4.7 Preconditioned Conjugate Gradient for solving linear system (3.4.20)

Inputs: $\boldsymbol{m}^{FWI}$, $\boldsymbol{m}'$, $\mathbf{u}(\boldsymbol{m}^{FWI})$, $\boldsymbol{\lambda}(\boldsymbol{m}^{FWI})$, $\mathcal{S}$, g, μ, α, L, $R(\boldsymbol{p})$, $\mathbf{x}_0$, tol_3,
Compute right hand side of (3.4.20) $\mathbf{e} = \boldsymbol{m}' - \boldsymbol{m}^{FWI}$
$H(\boldsymbol{m}^{FWI})\mathbf{x}_0 \leftarrow$ `Algorithm 5.4.8`$(\mathbf{x}_0, \boldsymbol{m}^{FWI}, \mathbf{u}(\boldsymbol{m}^{FWI}), \boldsymbol{\lambda}(\boldsymbol{m}^{FWI}), \mathcal{S}, g, \mu, \alpha, R(\boldsymbol{p}))$
$\mathbf{r}_0 \leftarrow \mathbf{e} - H(\boldsymbol{m}^{FWI})\mathbf{x}_0$
$\mathbf{z}_0 \leftarrow L^{-1}\left((L^T)^{-1}\mathbf{r}_0\right)$
$\boldsymbol{\rho}_0 \leftarrow \mathbf{z}_0$
$k = 0$
for $k = 1 : \text{length}(\mathbf{b})$ **do**
    $H(\boldsymbol{m}^{FWI})\boldsymbol{\rho}_k \leftarrow$ `Algorithm 5.4.8`$(\boldsymbol{\rho}_k, \boldsymbol{m}^{FWI}, \mathbf{u}(\boldsymbol{m}^{FWI}), \boldsymbol{\lambda}(\boldsymbol{m}^{FWI}), \mathcal{S}, g, \mu, \alpha, R(\boldsymbol{p}))$
    $a_k \leftarrow \dfrac{\mathbf{r}^T\mathbf{z}_k}{\boldsymbol{\rho}_k H(\boldsymbol{m}^{FWI})\boldsymbol{\rho}_k}$
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + a_k\boldsymbol{\rho}_k$
    $\mathbf{r}_{k+1} \leftarrow \mathbf{r}_k - a_k H(\boldsymbol{m}^{FWI})\boldsymbol{\rho}_k$
    if $\dfrac{||r_{k+1}||_2}{||r_0||_2} \leq tol_3$
        exit
    end if
    $\mathbf{z}_{k+1} \leftarrow L^{-1}\left((L^T)^{-1}\mathbf{r}_{k+1}\right)$
    $\beta_k \leftarrow \dfrac{\mathbf{r}_{k+1}^T\mathbf{z}_{k+1}}{\mathbf{r}_k^T\mathbf{z}_k}$
    $\boldsymbol{\rho}_{k+1} \leftarrow \mathbf{z}_{k+1} + \beta_k\boldsymbol{\rho}_k$
end for
$\boldsymbol{\delta} = \mathbf{x}_{k+1}$
Output: Solution to linear system $\boldsymbol{\delta}$

Algorithm 5.4.8 Hessian-vector products

Inputs: Vector $\boldsymbol{\rho}$, $\boldsymbol{m}^{FWI}$, $\mathbf{u}(\boldsymbol{m}^{FWI})$, $\boldsymbol{\lambda}(\boldsymbol{m}^{FWI})$, $\mathcal{S}$, g, μ, α, $R(\boldsymbol{p})$
Initialise $H\boldsymbol{\rho} = \left(\alpha D^T D + \mu I\right)\boldsymbol{\rho}$
for $s \in \mathcal{S}$
    for $k \in \{1, \ldots, N_g\}$
        $\mathbf{v} \leftarrow$ (2.4.22) evaluated at $\boldsymbol{m}^{FWI}$
        $\mathbf{z} \leftarrow$ (2.4.23) evaluated at $\boldsymbol{m}^{FWI}$
        for $j \in \{1, \ldots, M\}$
            $(H'\boldsymbol{\rho})_j \leftarrow$ (2.4.21) evaluated at $\boldsymbol{m}^{FWI}$
        end for
        $H\boldsymbol{\rho} \leftarrow H\boldsymbol{\rho} + H'\boldsymbol{\rho}$
    end for
end for
Output: Hessian vector product $H\boldsymbol{\rho}$ evaluated at $\boldsymbol{m}^{FWI}$

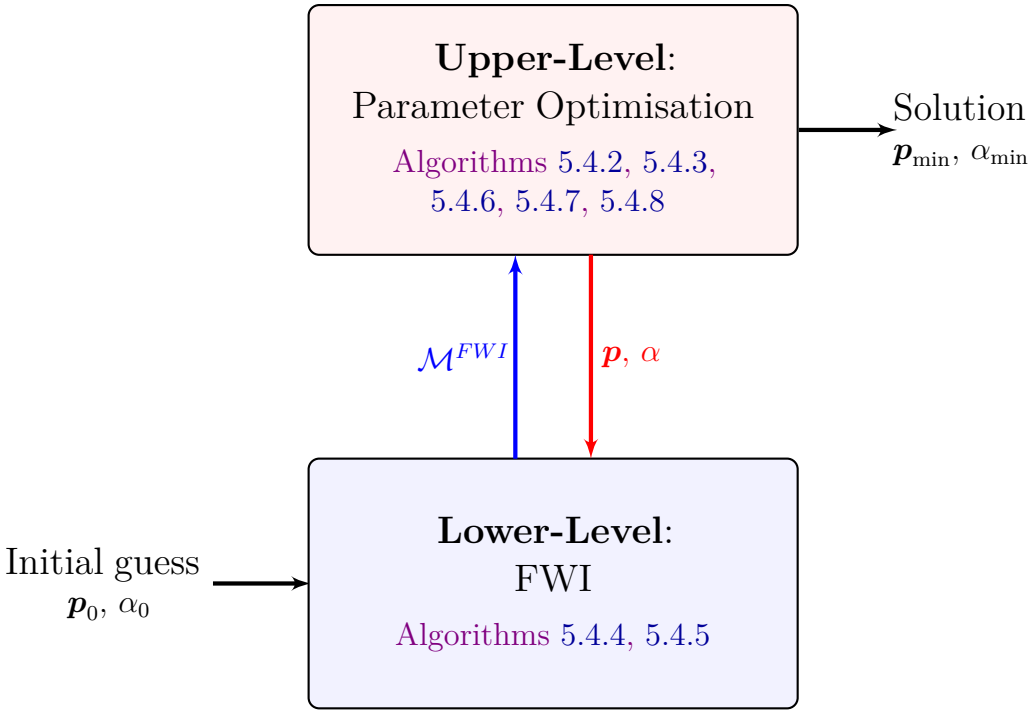


Figure 5.4.1: *Overall Schematic of the Bilevel Problem*

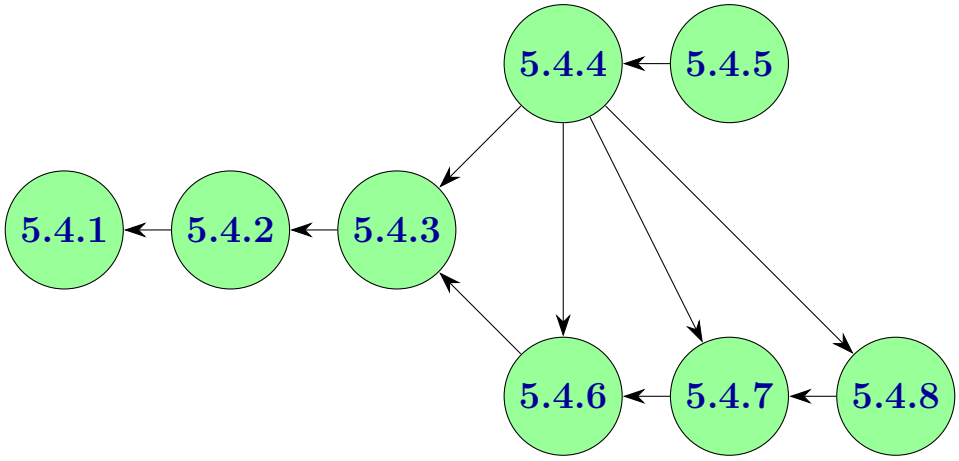
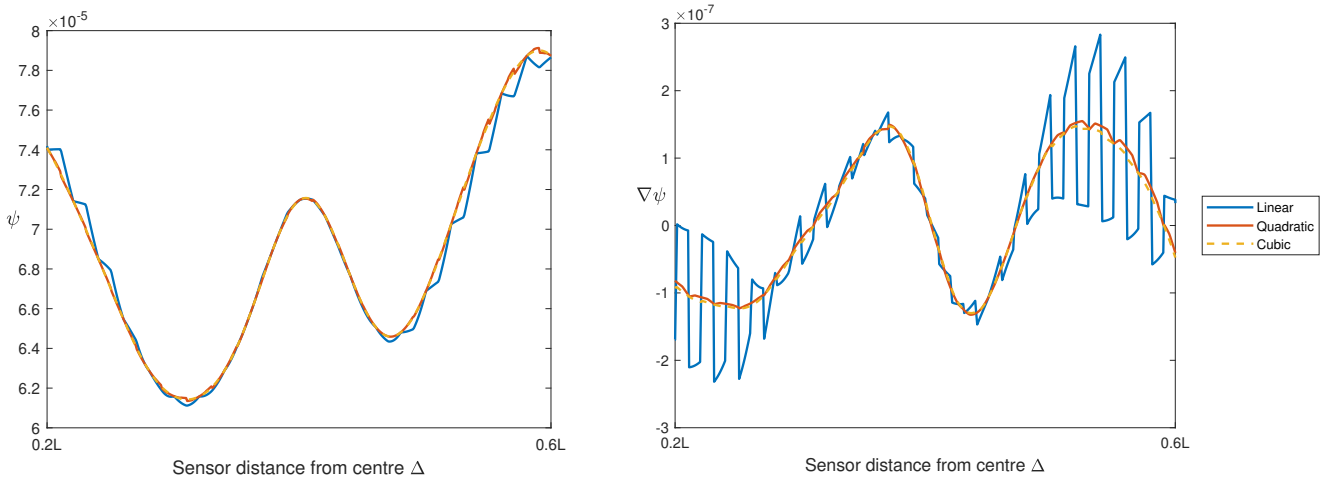


Figure 5.4.2: *Algorithm Dependency Graph: Arrows indicate dependency of one algorithm on another. The algorithm which the arrow is pointing to, for example Algorithm 5.4.6, depends on the algorithm that the arrow is coming from, either because that algorithm is called within it (e.g., Algorithm 5.4.7) or the outputs of that algorithm are inputs to it (e.g., Algorithm 5.4.4).*

Implementation of $R(\boldsymbol{p})$: In the algorithms presented here, we compute and use the restriction operator, and it's derivative with respect to sensor position. Although any interpolant can be chosen in the standalone FWI problem, as previously mentioned in Remark 3.4.31, the chosen interpolant for the bilevel problem should be 'smooth enough' such that the upper-level objective function is smooth and can be optimised by a local optimisation method.

We emphasise the importance of choosing the correct interpolant with the following example. We plot ψ and $\nabla\psi$ versus sensor position for the problem setup shown in Figure 5.1.1 (i.e., where sensors are moved along a line). We draw three different plots, one each for three different choices of $R(\boldsymbol{p})$ - one where $R(\boldsymbol{p})$ performs piecewise linear interpolation, one where $R(\boldsymbol{p})$ performs 'sliding' quadratic interpolation and one where $R(\boldsymbol{p})$ performs 'sliding' cubic interpolation (we explain how this is computed in the next paragraph). These are plotted for a group of two frequencies (4 and 5 Hz). In Figure 5.4.3, we focus in on the values of ψ and $\nabla\psi$ for sensor positions varied along a portion of the line in Figure 5.1.1 to demonstrate the differences in the plots computed with different interpolants. The ψ plot computed using a piecewise linear interpolant is not smooth in several places, and the $\nabla\psi$ plot has large sharp discontinuities. These discontinuities are due to the fact that the derivative of a piecewise linear interpolant is discontinuous. The plots drawn using a quadratic interpolant appear a lot smoother, although looking at the right side of the $\nabla\psi$ plot shows some small bumps. In contrast the plots drawn using the cubic interpolant appears smooth everywhere.



(a) ψ (b) $\nabla\psi$

Figure 5.4.3: *Plots of ψ and $\nabla\psi$ usuing different implementations of $R(\boldsymbol{p})$.*

Due to the smoothness we have observed for the 'sliding' cubic interpolant, we use a 'sliding' bicubic interpolant in our implementation. In one dimension, for each sensor (i.e., for each row of $R(\boldsymbol{p})$), this involves finding the four grid points that are closest to the sensor position, and performing cubic interpolation on those gridpoints. In two dimensions, to compute each row of $R(\boldsymbol{p})$, we find the four closest gridpoints in each direction, find the cubic interpolant in both directions separately, and then compute the

tensor product of these two cubic interpolants to get the bicubic interpolant. Therefore, when we apply $R(\boldsymbol{p})$ to the wavefield $\mathbf{u}$, we are only using wavefield information at the gridpoints. When the sensor position moves into a different interval, we just find the four closest gridpoints to this new interval and repeat the interpolation.

Since $dR(\boldsymbol{p})/dp_n$ is extremely sparse, and only has non-zero values on row n, we only need to compute and store the vector $d\left(R(\boldsymbol{p})\right)_n/dp_n$. Therefore in the implementation we compute $d\left(R(\boldsymbol{p})\right)_n/dp_n$ for all n at once and store the results together in one $dN_r \times N$ matrix.

Remark 5.4.2. *The algorithms in this section present the case where both sensor positions $\boldsymbol{p}$ and the Tikhonov parameter α are optimised together in every frequency group. As discussed in Section 5.1, other approaches to bilevel frequency continuation involve optimising $\boldsymbol{p}$ alone in some groups before including α in the optimisation. The real implementation used in experiments in this thesis is slightly more complicated than the algorithms written down here, as it allows the option to optimise $\boldsymbol{p}$ alone or simultaneously with α, depending on the frequency group. The implementation allows α to start being optimised in any frequency group, depending on the user's choice. We have avoided incorporating the option to start optimising α in different frequency groups into the written algorithms in this section to make the algorithms more readable, but we briefly describe here how one can implement the more flexible approach here.*

We introduce the notation N_α, such that g_{N_α} is the frequency group in which we start to optimise the Tikhonov parameter α. The user inputs their choice of N_α. Up to the N_αth frequency group, $\boldsymbol{p}$ is optimised alone, and α is treated as a constant, and once we get to the frequency group of choice, both $\boldsymbol{p}$ and α are optimised together. In Algorithm 5.4.9, we input the current frequency group number, k_g, and the frequency group at which we start to include α in the optimisation, N_α, into all the algorithms that are called within Algorithm 5.4.9. The group number information tells the other algorithms what parameters are being optimised, and hence how to compute quantities such as the upper-level gradient. We note that there are other ways that the information on what parameters are being optimised can be accessed by the other algorithms, for example by including a flag when α is also being optimised, or by getting the algorithm to check if the sizes of $\boldsymbol{p}$ and $\boldsymbol{q}$ are the same.

Remark 5.4.3. *In the real implementation, there is also a choice of how many dimensions the sensors should be optimised in. The user inputs the direction of optimisation, i.e., either the sensors should be optimised in the x, z or both directions, and the following algorithms all account for this. The algorithms written in this section just describe the case where all sensor coordinates are optimised.*

Algorithm 5.4.9 Bilevel Frequency Continuation with Choice of Optimisation Parameters

Inputs: $\boldsymbol{p}_0$, $\mathcal{M}_0$, $\{\omega_1 < \omega_2 < ... < \omega_{N\omega}\} \in \mathcal{W}$, $\mathcal{M}'$, α_0, μ, N_α
Compute preconditioner $\mathbf{P}$ with α_0 and μ
$L \leftarrow \text{Cholesky}(\mathbf{P})$
Group frequencies into N_f groups $\{g_1, g_2, \ldots, g_{N_f}\}$
for $k_g = 1$ **to** N_f **do**
    if $k_g < N_\alpha$
        $[\boldsymbol{p}_{\min}, \mathcal{M}^{FWI}] \leftarrow$ `Algorithm 5.4.2`$(g_k, \boldsymbol{p}_0, \alpha_0, \mathcal{M}_0, \mathcal{M}', L, k_g, N_\alpha)$
        $\boldsymbol{p}_0 \leftarrow \boldsymbol{p}_{\min}$
        $\mathcal{M}_0 \leftarrow \mathcal{M}^{FWI}$
    else
        $[\boldsymbol{p}_{\min}, \alpha_{\min}, \mathcal{M}^{FWI}] \leftarrow$ `Algorithm 5.4.2`$(g_k, [\boldsymbol{p}_0; \alpha_0], \mathcal{M}_0, \mathcal{M}', L, k_g, N_\alpha)$
        $\boldsymbol{p}_0 \leftarrow \boldsymbol{p}_{\min}$
        $\alpha_0 \leftarrow \alpha_{\min}$
        $\mathcal{M}_0 \leftarrow \mathcal{M}^{FWI}$
    end if
end for
Output: $\boldsymbol{p}_{\min}$, $\alpha_{\min}$

Chapter 6

Large Scale Parameter Optimisation Experiments

Chapter Summary: Throughout this thesis, various aspects of FWI and the parameter optimisation problems have been demonstrated through numerical illustrations. In this chapter, we present some further parameter optimisation problems, and we test and comment on the results.

In Experiment 1 (§6.1), we apply our bilevel learning algorithm to a training set of several distinct models that share some common characteristics. We test the resulting optimal parameters extensively on randomly generated testing sets with various different properties to investigate how well our optimal parameter results generalise. In Experiment 2 (§6.2), we optimise the parameters of a more difficult problem, the Marmousi model, showing that our bilevel learning algorithm can be applied successfully to realistic geophysical subsurfaces.

6.1 Experiment 1

The aim of Experiment 1 is to test how well the results of the parameter optimisation generalise to models that are not in the training set. We define the term *class of models* here to mean a set of models that are similar in some way, i.e., they all share certain properties. We choose a training set of models that are all in the same class, learn the optimal sensor positions and Tikhonov regularisation parameter, and then test the optimised parameters on models within and outside of this class.

6.1.1 Training

Experiment Details: The training set that we choose is shown in Figure 6.1.1. This set involves twelve models that are all in the same class - they all feature two smooth Gaussian bumps of equal wavespeed, surrounded by a constant lower wavespeed, and positioned along the diagonal of the domain. Half of the training models involve bumps that are equal in size to each other, and the other half involves bumps which differ in size to each other. The maximum wavespeed at the centre of the bump is varied across

the training models, in the range 2.1 km/s to 2.2 km/s. The background wavespeed is constant at 2 km/s, meaning that the 'height' of the bump is varied between 100 m/s and 200 m/s. The radius of the bumps is varied between 150 and 250 m. The positions of the bumps vary along the diagonal but the bumps do not overlap significantly and do not significantly extend beyond the border of our domain (which we ensure by choosing the radius and position carefully).

The bilevel algorithm (Algorithm 5.4.9) is applied to this training set to learn the optimal positions $\boldsymbol{p}$ for ten sensors and the optimal Tikhonov regularisation parameter α. We recall from Remark 3.3.3 that no noise is added to the training data. The initial guess for FWI is a model with a constant wavespeed of 2 km/s. We use three frequency groups, with frequencies in the range 0.5 Hz to 2.5 Hz, and optimise α in the last group only. The initial guess for the regularisation parameter is $\alpha_0 = 0.1$ and the initial guess for the ten sensor positions is uniform placement along a vertical line. There are ten sources, which are placed uniformly along a vertical line on the left side of the domain. The lower-level problem is solved to a tolerance of $||\nabla\phi||_2 \leq 10^{-10}$. The linear system involved in the upper-level gradient computation is solved to a tolerance of 10^{-15}, to ensure the error in the gradient is as low as possible. Since the upper-level optimisation method we are using is a bounded algorithm (L-BFGSb), convergence of the upper-level problem is achieved when the infinity norm of the projected gradient is smaller than some tolerance ($\leq 10^{-10}$), or alternatively when the updates to ψ or the optimisation parameters stall, or the maximum number of iterations is reached (which is 50 iterations for each frequency group). We briefly note that the projected gradient is the gradient that has been projected onto the feasible region, and is defined explicilty in [39].

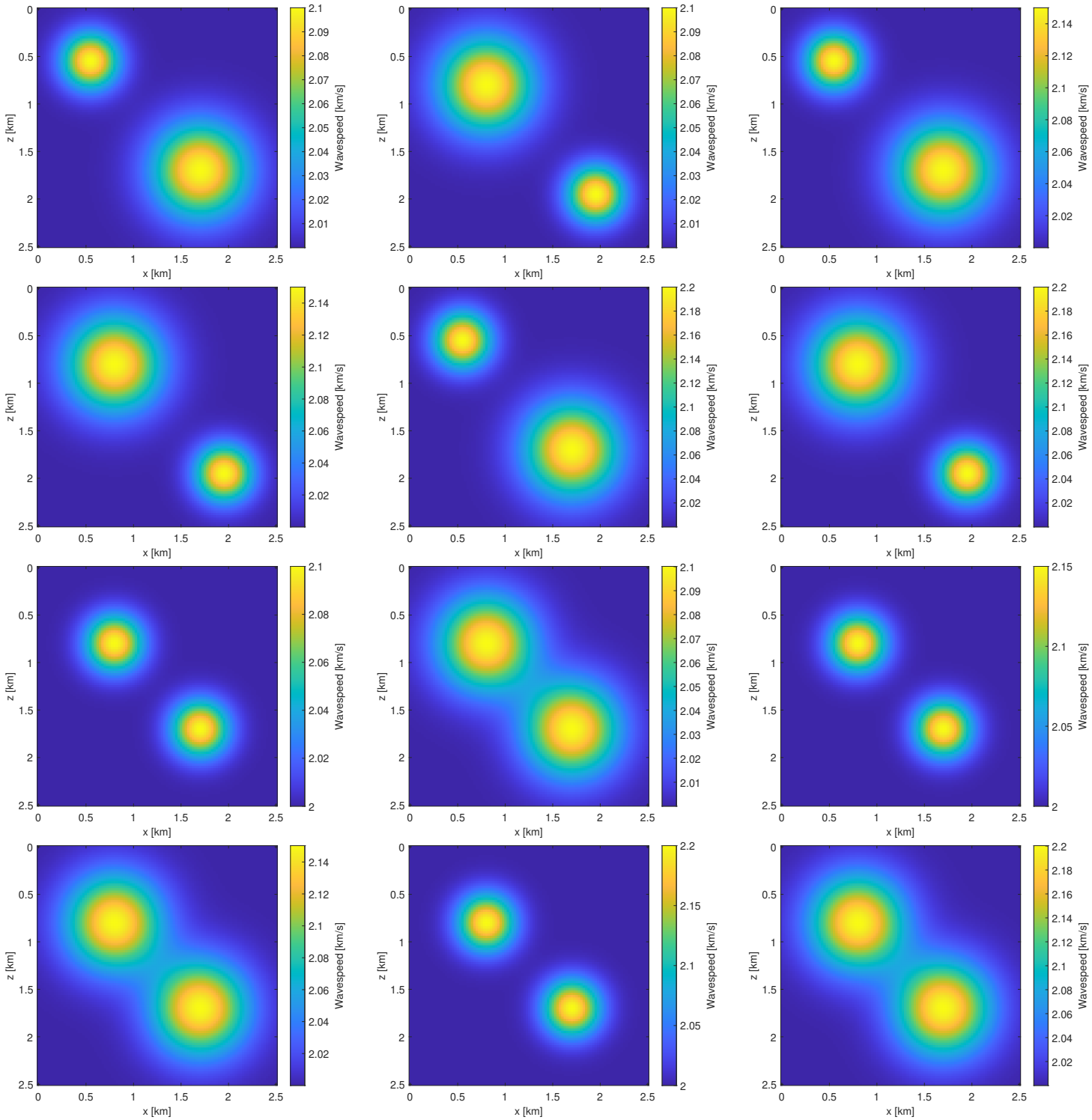


Figure 6.1.1: *Training set used for Experiment 1.*

Results: The initial and optimal setup is overlayed on one of the training models in Figure 6.1.2. We observe that the sensors spread out to get better coverage of the domain, and that five of the sensors seem to line up along the diagonal that the Gaussian bumps are positioned on. We also notice that four of the sensors are positioned along the right side of the domain, opposite from the sources, allowing the sensors to record waves transmitted through the whole domain.

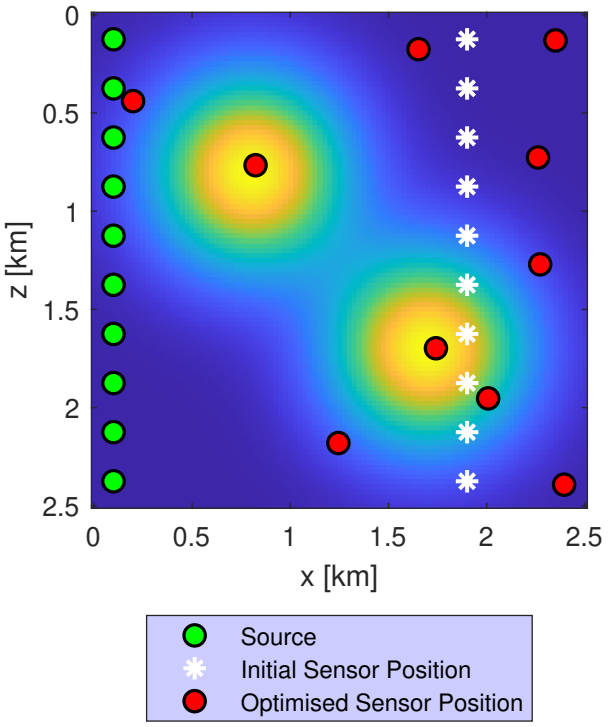
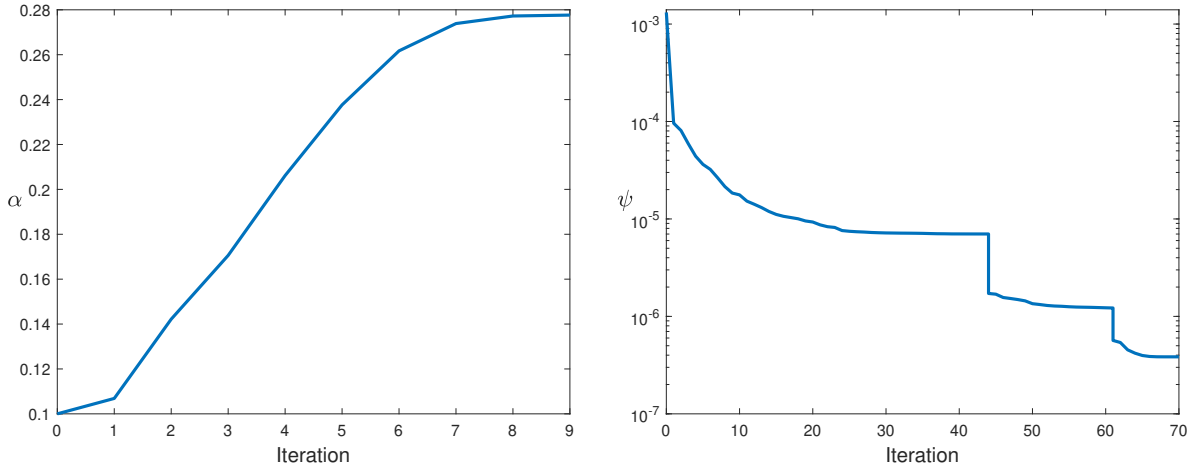


Figure 6.1.2: *Initial and optimal sensor positions for the training set in Figure 6.1.1.*

Figure 6.1.3 (a) is a plot of α versus iteration in the final group. We see that α grows slowly and converges at approximately $\alpha_{\min} = 0.278$. Figure 6.1.3 (b) demonstrates the decrease in ψ versus iteration. The first frequency group takes the longest to converge (44 iterations) as the sensors make the largest changes in their positions in this group. The final group converges the quickest as the sensors begin the group very close to their optimal positions and only the regularisation parameter needs to change. In addition, the starting guess for the regularisation parameter is relatively close to its optimal value. A drop in ψ is evident as we progress from one group to the next (at iteration 44 and iteration 61).



(a) Tikhonov regularisation parameter α (b) Objective function ψ

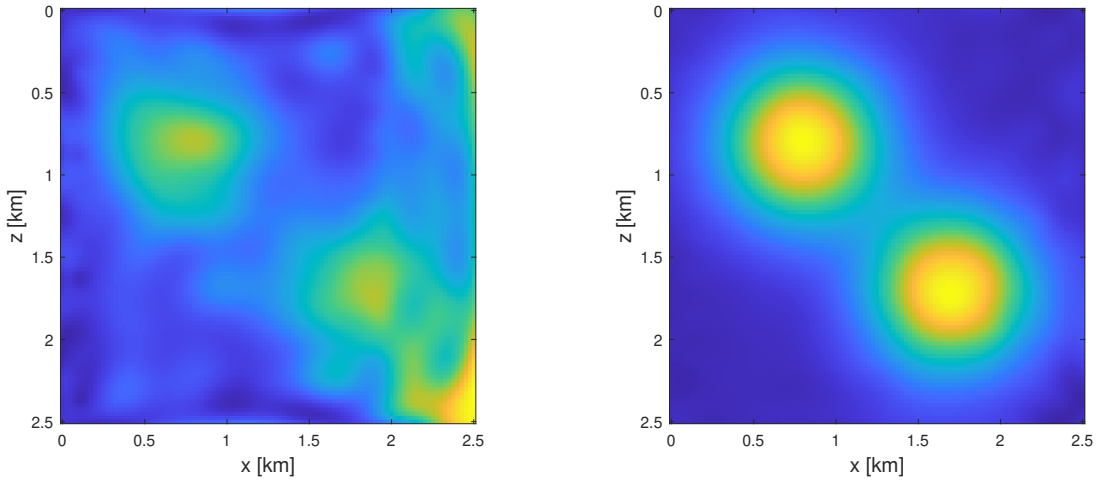
Figure 6.1.3: *Variation in the Tikhonov regularisation parameter α (in the final frequency group) and the objective function ψ verus iteration.*

In Figure 6.1.4 we show an example of the improvement in the FWI reconstruction of the training model shown in Figure 6.1.2 after the bilevel algorithm has been applied. Subfigure (a) shows the reconstruction at the initial sensor positions and regularisation parameter, and subfigure (b) shows the reconstruction at the optimal parameters. Visually, there is a large increase in the quality of the reconstruction between (a) and (b). In terms of the value of ψ, the improvement factor (defined in (3.6.1)) for this training model is 878 (rounded to the nearest integer). This training model had the largest improvement factor of all training models. A summary of the improvement factors across the whole training set is contained in Table 6.1.1, and we plot the improvement factors for each training model in Figure 6.1.5. The models in Figure 6.1.5 are ordered according to how they appear in Figure 6.1.1. We can see from this plot that the largest improvement factors are reported for the training models with two large Gaussian bumps, and that in general, the training models with smaller Gaussian bumps and a higher maximum wavespeed have lower improvement factors.

Improvement Factor		
Minimum	Maximum	Average
240	878	482

Table 6.1.1: *Summary of improvement factors found for Testing Set 1.*

As an alternative measure of the quality of reconstruction, we compute the relative percentage error (2.6.2) at every point of the model, and find the average across all the points. We compute this value for all training models after the bilevel algorithm has been applied. Across all training models, the range of average relative errors is from 0.0295% to 0.0655%, with an average of 0.0470%.



(a) $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \alpha_0)$ (b) $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$

Figure 6.1.4: *FWI reconstruction of the training model in Figure 6.1.2, at the initial guess at parameters* $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \alpha_0)$ *and optimised parameters* $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$.

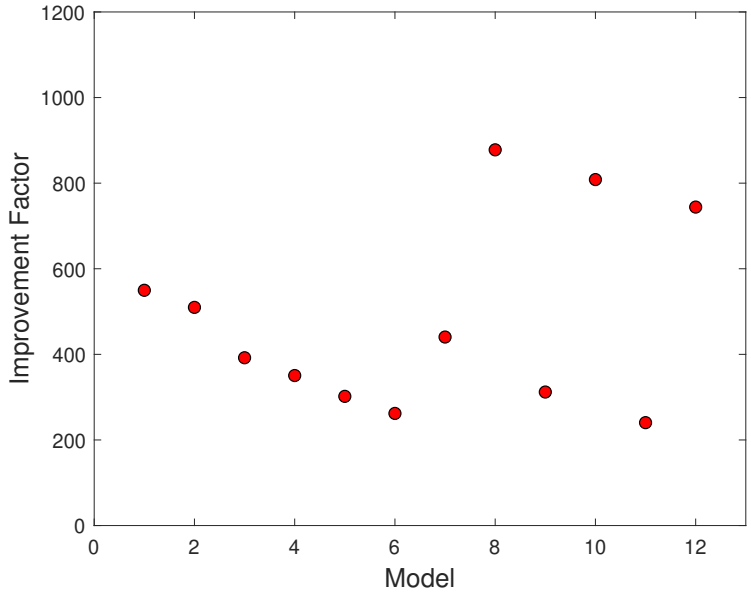


Figure 6.1.5: *Improvement factors for the training set.*

6.1.2 Testing

In order to assess the performance of the optimal parameters obtained by our bilevel algorithm, we generate random testing sets of models, perform FWI using the initial and optimised parameters on these testing models, and compute the improvement factors. We test on different testing sets, both within the same class and in different classes as the training set. All testing sets involve Gaussian bumps so that there is some relation between the training and testing set. If the improvement factors remain high for all

testing sets, then we know that the solution of our bilevel algorithm generalises well to models outside of the training set. We also report the average relative percentage error across each testing set to provide a measure of quality that is independent of the starting guess.

Testing Set 1: The first testing set involves 50 randomly generated models in the same class as the training set, i.e., all testing models involve two Gaussian bumps along the diagonal, and the size and height of the bumps are within the same range as those in the training set. A subset of these testing models are shown in Figure 6.1.6. In the random generation of models, care is taken to ensure that bumps do not overlap significantly or that parts of the bump do not extend beyond the border of our domain.

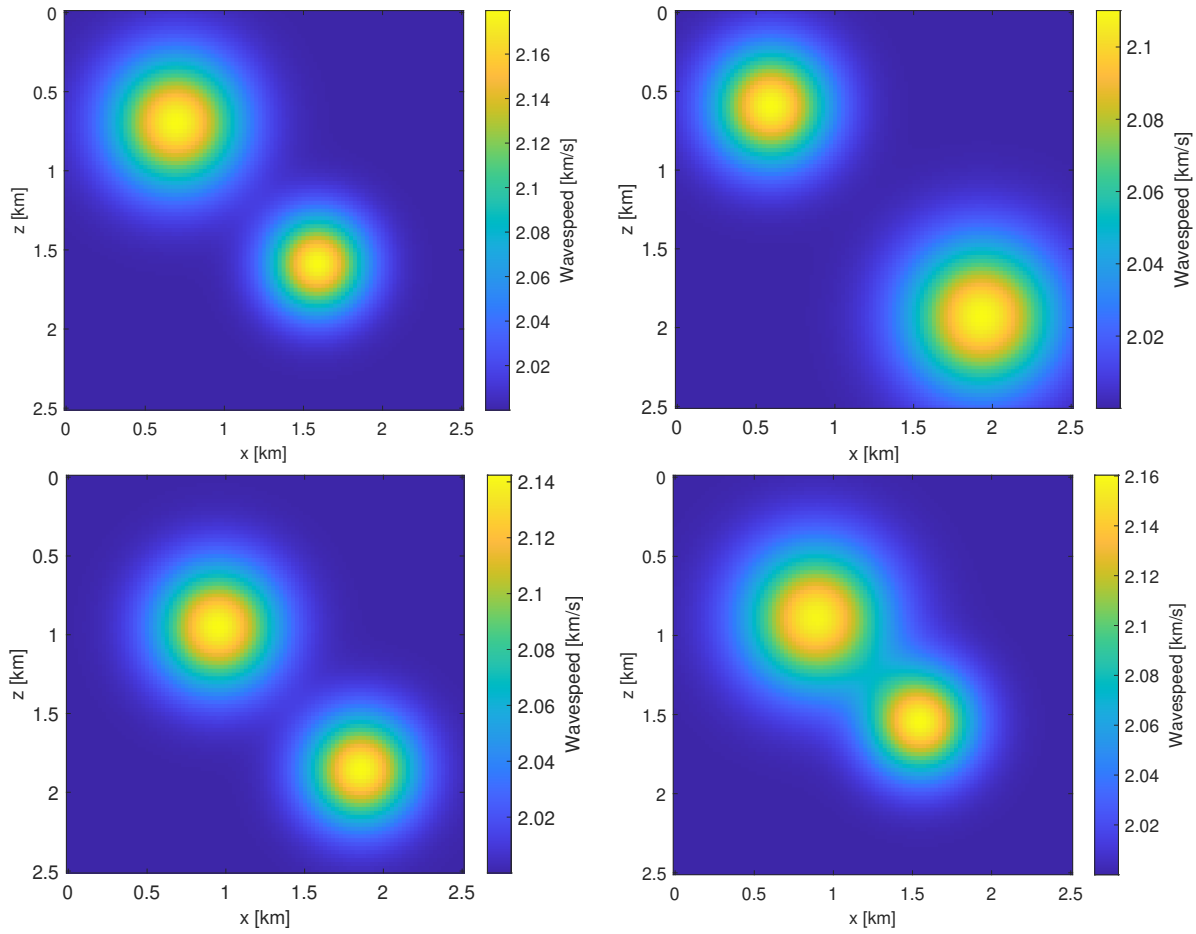


Figure 6.1.6: *Subset of Testing Set 1.*

At the optimal parameters, the average relative percentage error across all testing models in this set is 0.0445%, showing that all testing models are reconstructed accurately with our optimal parameters. The improvement factors for all 50 testing models are displayed in Figure 6.1.7, and the maximum, minimum and average improvement factors found for this testing set of 50 models are shown in Table 6.1.2, where values are rounded

to the nearest integer. The average improvement factor is actually larger for this testing set than the training set (which may just be related to the fact that the testing set is larger), but the range in improvement factors remains very similar. We note that the larger improvement factors come from the models with two larger Gaussian bumps, and the smaller improvement factors come from the testing models with two small Gaussian bumps.

Improvement Factor		
Minimum	Maximum	Average
287	842	536

Table 6.1.2: *Summary of improvement factors found for Testing Set 1.*

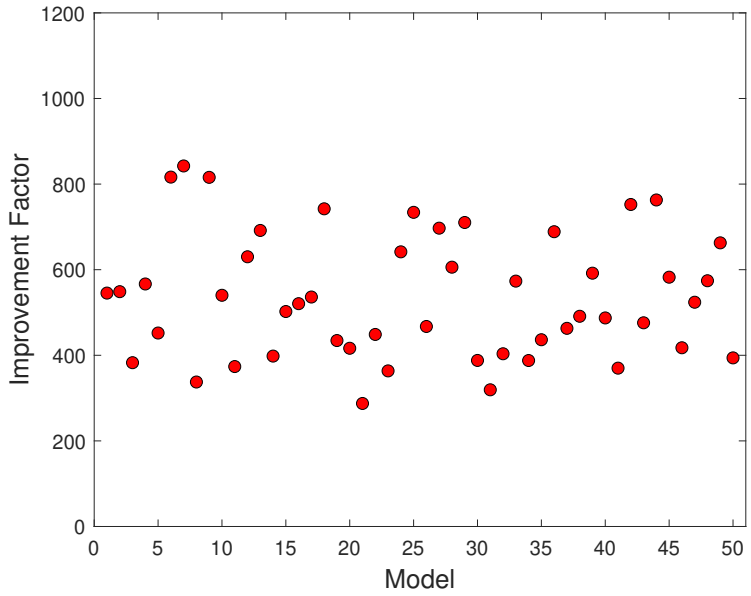
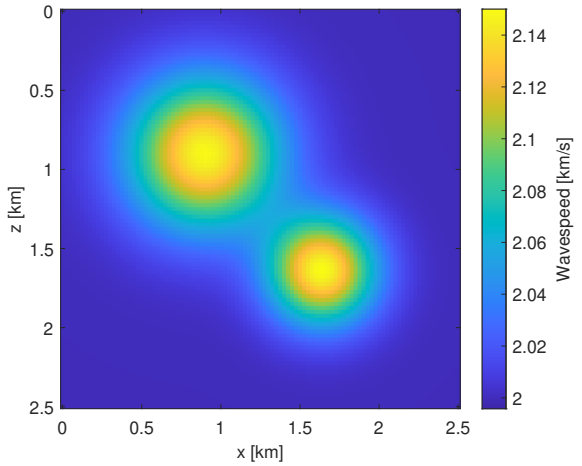


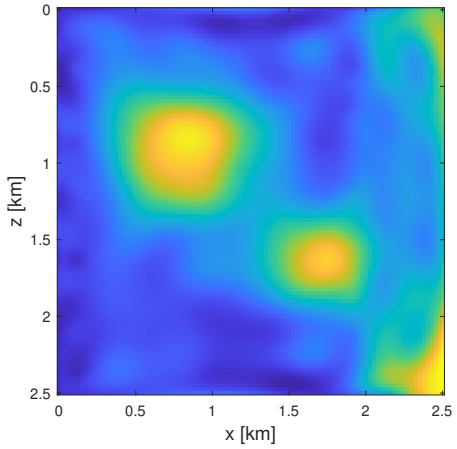
Figure 6.1.7: *Improvement factors for Testing Set 1.*

In Figure 6.1.8 we include an example of the improvement in FWI reconstruction for one of the models in the testing set where the improvement factor is close to the average value.

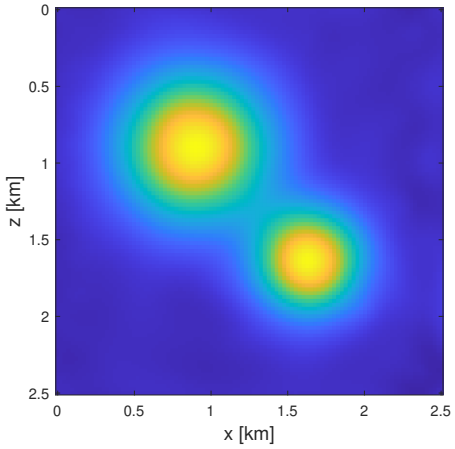
We conclude from Testing Set 1 that the results of the bilevel algorithm generalise well to models that are not in the training set but are in the same class as the training set.



(a) Testing model ground truth



(b) $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \alpha_0)$



(c) $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$

Figure 6.1.8: *Example of a testing model in Testing Set 1 and the corresponding FWI reconstructions at the initial parameters* $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \alpha_0)$*, and optimised parameters* $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$.

Testing Set 2: The next testing set is made up of 50 randomly generated models in a different class to the training set. Testing set 2 still involves two Gaussian bumps, with a height and size in the same range as the testing set, however, the positions of the bumps are now along the opposite diagonal. A subset of these models are shown in Figure 6.1.9.

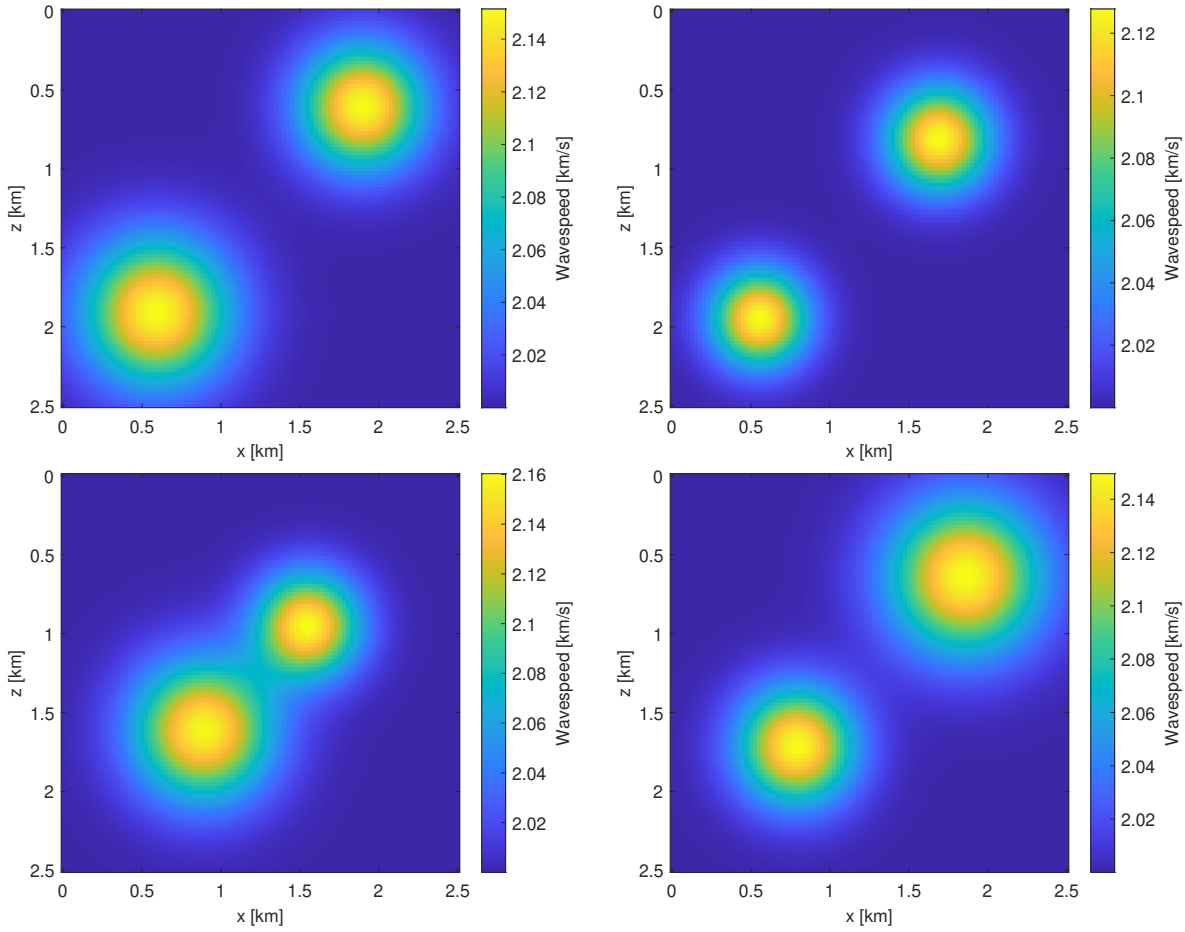


Figure 6.1.9: *Subset of Testing Set 2.*

The average relative percentage error across Testing Set 2 is 0.0671%, which remains low but is an increase from Testing Set 1. The improvement factors for all 50 testing models are shown in Figure 6.1.10, and summary statistics of the improvement factors (rounded to the nearest integer) are displayed in Table 6.1.3. The average has dropped from the last testing set, and the minimum has reduced by over three times. This reduction in improvement from Testing Set 1 is probably due to some of the optimal sensor positions being aligned along the top left to bottom right diagonal. The sensor positions are therefore not positioned to be optimal for Gaussian bumps positioned along the opposite diagonal. The largest improvement factors correspond to testing models with large Gaussian bumps and the smallest improvement factors correspond to testing models with smaller Gaussian bumps, particularly when one of the bumps is located near the bottom left of the domain where there are no sensors.

Although the improvement factors have reduced, there is still a consistent and large improvement across all testing models, showing that the optimal sensor positions found using a training set in a certain class generalises well to a testing set that is of a different class, but shares some properties with the training set.

Improvement Factor		
Minimum	Maximum	Average
93	663	307

Table 6.1.3: *Summary of improvement factors found for Testing Set 2.*

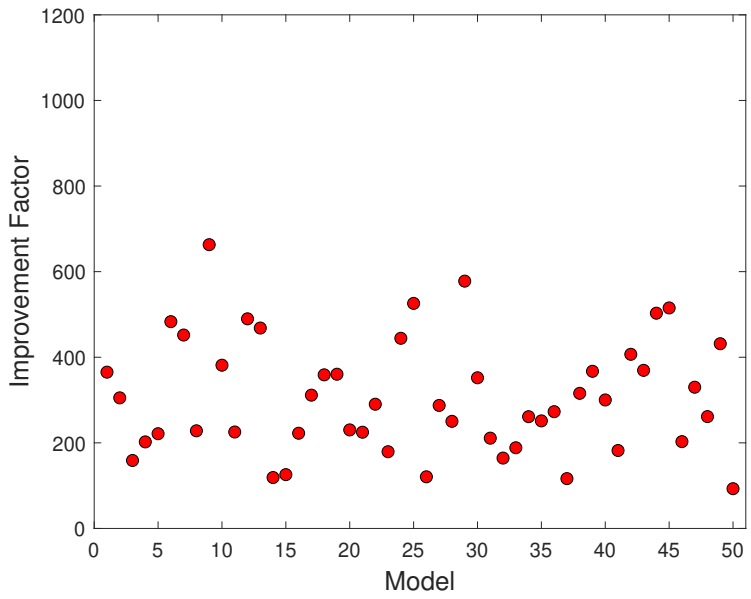
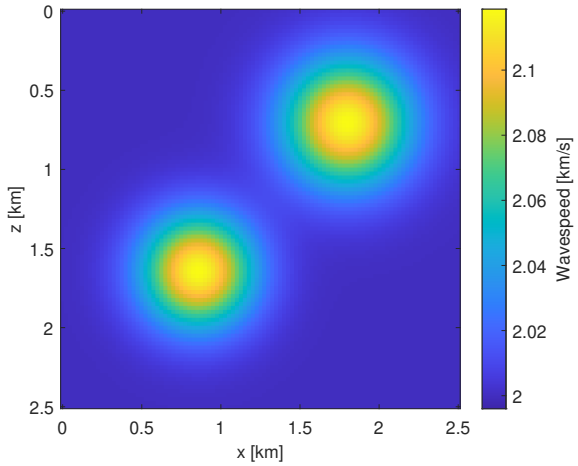
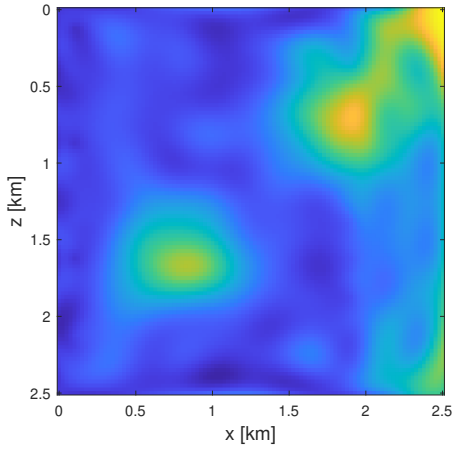


Figure 6.1.10: *Improvement factors for Testing Set 2.*

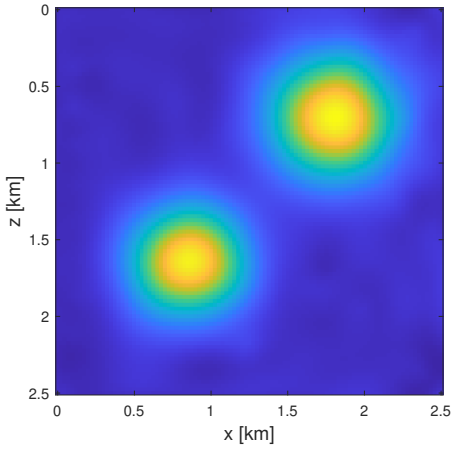
In Figure 6.1.11, we include an example of the FWI reconstructions of one of the testing models that has an improvement factor close to the average in Table 6.1.3.



(a) Testing model ground truth



(b) $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \alpha_0)$



(c) $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$

Figure 6.1.11: *Example of a testing model in Testing Set 2 and the corresponding FWI reconstructions at the initial parameters* $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \alpha_0)$*, and optimised parameters* $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$.

Testing Set 3: This testing set is again made up of 50 randomly generated models in a different class to the training set. This testing set involves only one Gaussian bump, but its height and size are all randomly generated from the same range as the training set, and the position of the bump is along the same diagonal as those in the training models. A subset of Testing Set 3 is shown in Figure 6.1.12.

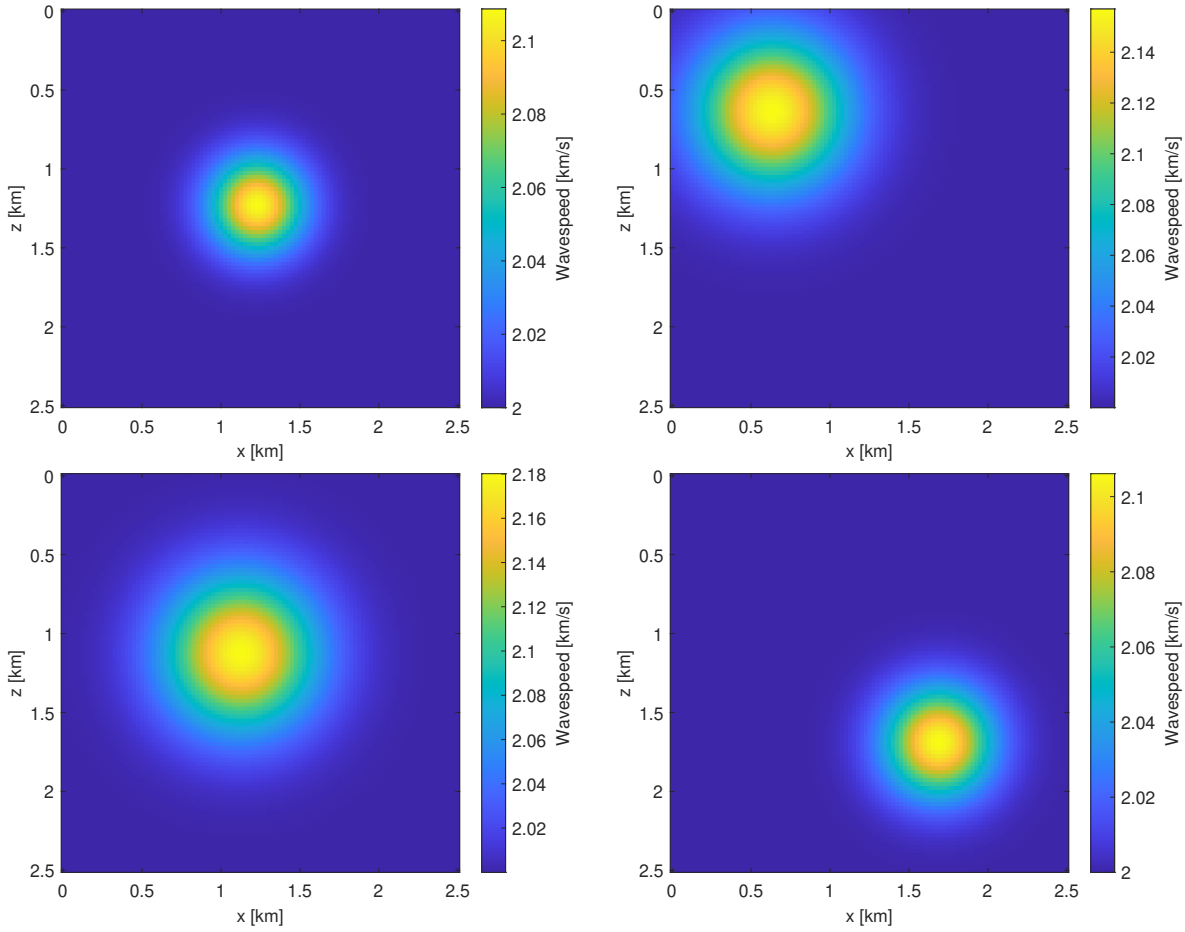


Figure 6.1.12: *Subset of Testing Set 3.*

The improvement factors for all models in Testing Set 3 are displayed in Figure 6.1.13, and the minimum, maximum and average improvement factors for this set are shown in Table 6.1.4. The improvement factors for this set are even larger than for Set 1 (which is in the same class as the training set). This is possibly because a model with only one Gaussian bump may be easier to reconstruct with 10 sensors than a model with two Gaussian bumps. This feature is also seen through the relative percentage error measure of quality, where the average is smaller than Set 1 (0.0389%).

Improvement Factor		
Minimum	Maximum	Average
417	964	725

Table 6.1.4: *Statistics of improvement factors found for Testing Set 3.*

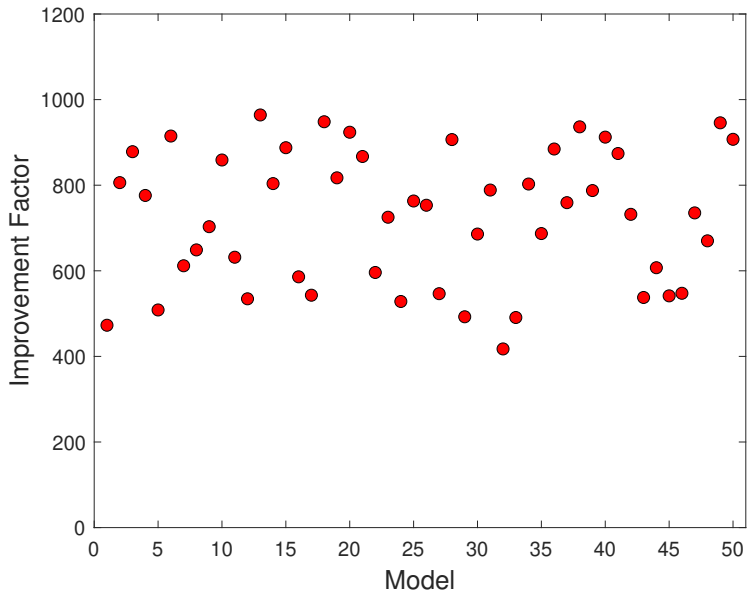
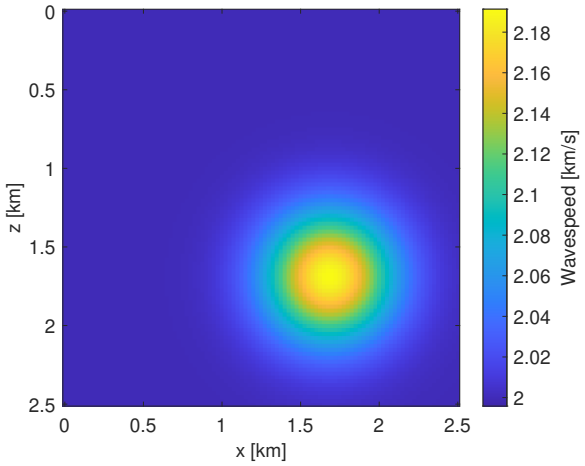


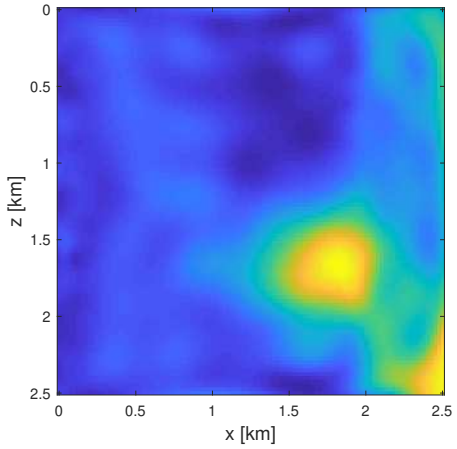
Figure 6.1.13: *Improvement factors for Testing Set 3.*

In Figure 6.1.14, we include an example of the FWI reconstructions of one of the testing models that has an improvement factor close to the average in Table 6.1.4.

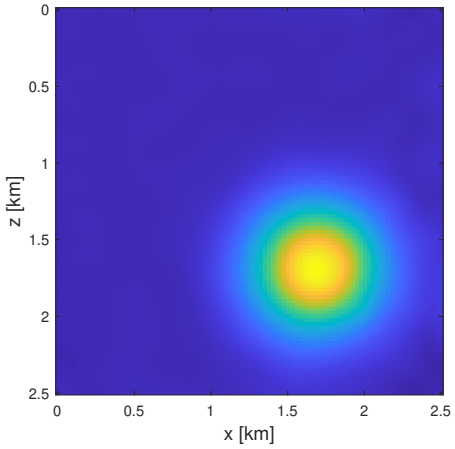
This testing set shows that it is possible that optimal parameters that are learned on training models of a certain class can also perform well, or even better, on models outside of this class.



(a) Testing model ground truth



(b) $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \alpha_0)$



(c) $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$

Figure 6.1.14: *Example of a testing model in Testing Set 3 and the corresponding FWI reconstructions at the initial parameters* $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \alpha_0)$*, and optimised parameters* $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$.

Testing Set 4: So far, the testing sets have either been in the same class as the training set or have only one property different from the training set. In Testing Set 4, we allow more properties to vary outside the ranges used in the training set. This testing set is made up of models with one Gaussian bump, with height randomly chosen within the range 100 m/s to 400 m/s, radius chosen within the range 100 m to 500 m, and position randomly chosen from the whole domain, i.e., it is not constrained to be on the diagonal and it is possible that parts of the bump extend beyond the domain. A subset of these randomly generated models are shown in Figure 6.1.15.

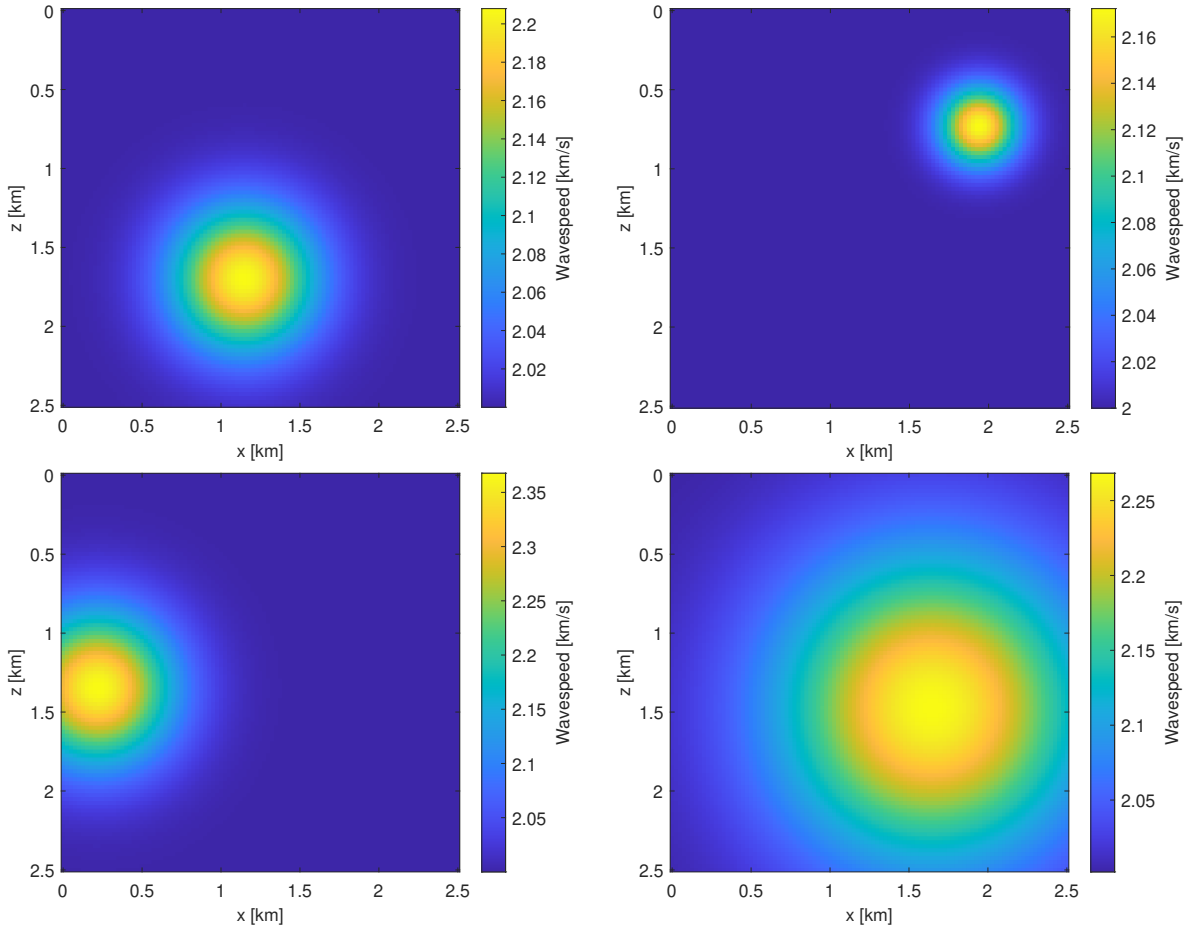


Figure 6.1.15: *Subset of Testing Set 4.*

Figure 6.1.16 and Table 6.1.5 report the improvement factors for this testing set. The range of improvement factors has approximately doubled from the previous tests. While the smallest improvement factor is only 42, the largest has increased to 1171. While an improvement factor of 42 is a decrease from the factors seen before, there is still a substantial improvement in the quality of the FWI reconstruction, and this shows that even when the testing model varies a lot from the training models, we are still seeing improvements through the use of the optimal parameters found by our bilevel algorithm. In addition, the average improvement factor remains large, and the average relative percentage error remains small (0.0505%).

Improvement Factor		
Minimum	Maximum	Average
42	1171	568

Table 6.1.5: *Statistics of improvement factors found for Testing Set 4.*

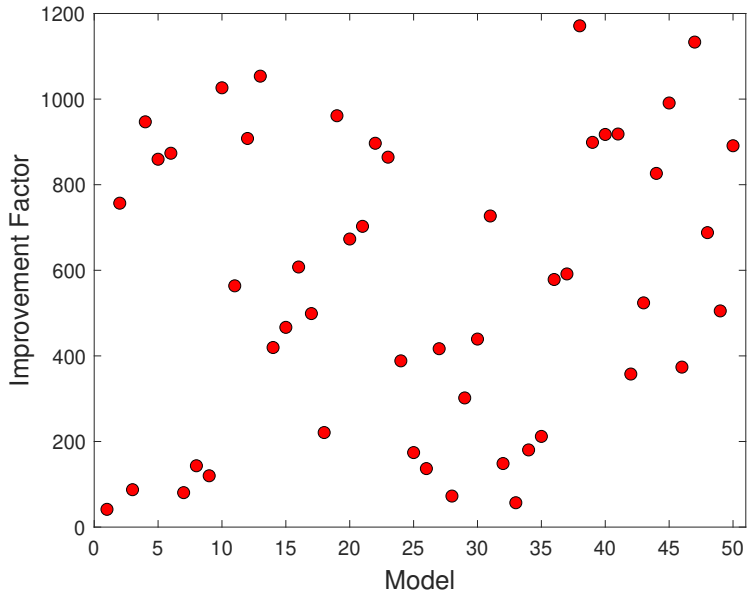
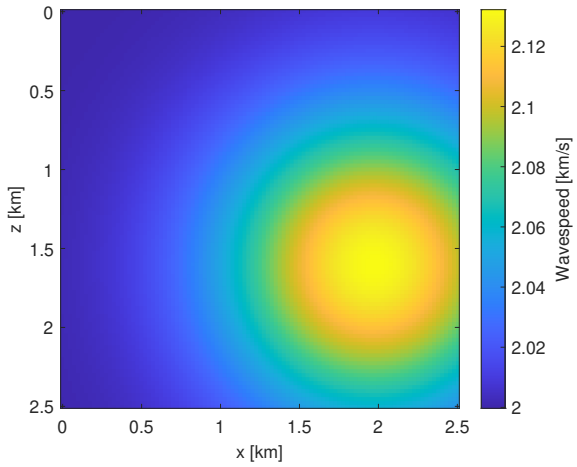


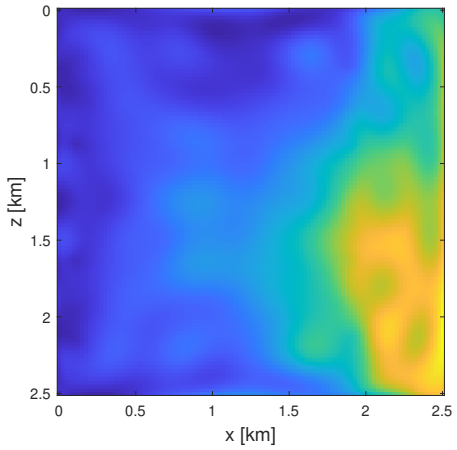
Figure 6.1.16: *Improvement factors for Testing Set 4.*

In Figure 6.1.17, we include an example of the FWI reconstructions of one of the testing models.

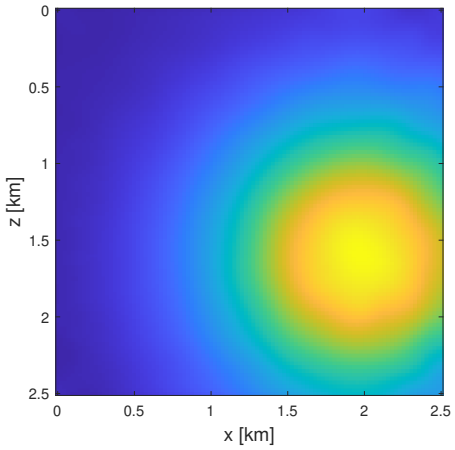
This testing set shows that the optimal parameters generalise well to models that vary considerably from the training set class. Although the improvement factor varies drastically between testing models here, there is a steady improvement and the potential for an extremely large improvement.



(a) Testing model ground truth



(b) $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \alpha_0)$



(c) $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$

Figure 6.1.17: *Example of a testing model in Testing Set 4 and the corresponding FWI reconstructions at the initial parameters* $\boldsymbol{m}^{FWI}(\boldsymbol{p}_0, \alpha_0)$*, and optimised parameters* $\boldsymbol{m}^{FWI}(\boldsymbol{p}_{\min}, \alpha_{\min})$.

Comparison of Testing Sets: The improvement factors for all testing sets are compared in Figure 6.1.18. The training set and Testing Set 1 have a similar range in improvement factors, which is logical since Testing Set 1 is in the same class as the training set. In general, the factors for Testing Set 1 is clustered about the centre of this plot, the factors for Set 2 are on average lower, and the factors for Set 3 are on average higher, but there is significant overlap between these three groups. The factors for Testing Set 4 vary greatly, and feature the overall maximum and minimum values.

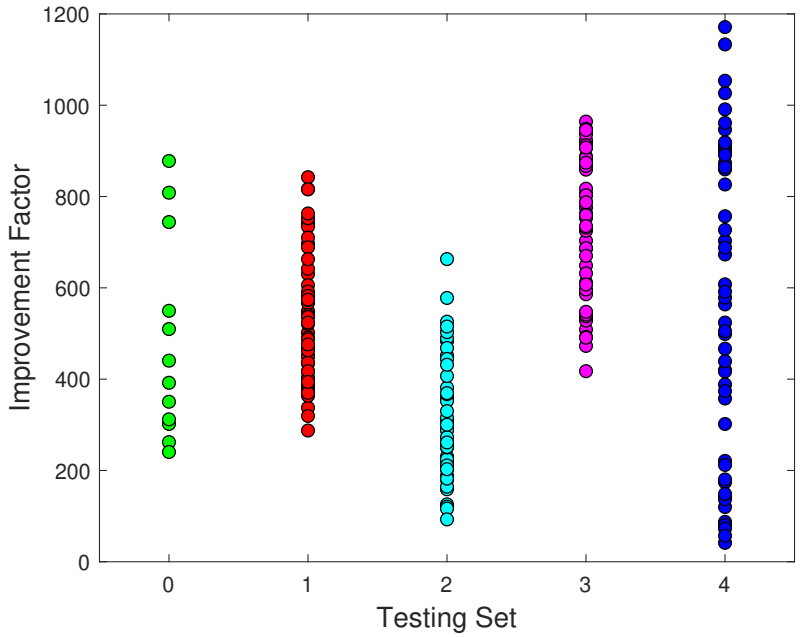


Figure 6.1.18: *Improvement factors for the training set (at 0) and all testing sets (1-4).*

Conclusion: To summarise, the testing in this section shows that the optimal parameters learned from a specific training set generalise well to models that are not in the training set, both within the same class and outside the class of the training set. These experiments also establish that we do not require a large training set to find optimal parameters that generalise well: indeed, we used only 12 training models here, and found that for all 200 randomly generated testing models, there was a consistent improvement in the quality of the reconstruction when using the optimal parameters found by our bilevel algorithm. Of course, we can't expect that the same sensor positions and regularisation parameters will be optimal for all classes of models, but we have shown that, as long as there are some features in common between the training and testing models (which in this case was that all models involved Gaussian bumps of higher wavespeed than the surrounding wavespeed), the optimal parameters have the ability to be applied successfully to a large range of models with varying properties. This makes the algorithm extremely useful, since, even though it is expensive, it would generally only needed to be performed once to determine optimal parameters that can then can be applied to a relatively wide range of situations.

6.2 Experiment 2

The aim of Experiment 2 is to apply our bilevel learning algorithm to find the optimal parameters for a realistic geophysical subsurface. We choose to apply our algorithm to the *smooth Marmousi* velocity model. The Marmousi model, introduced in Section 2.6, is a realistic geophysical subsurface section, and is a regularly used test case for

seismic imaging problems. For our purposes, the Marmousi model has been smoothed horizontally and vertically using a Gaussian filter. The smooth Marmousi model is shown in Figure 6.2.1. We note that, in general, the wavespeed in this model increases with depth.

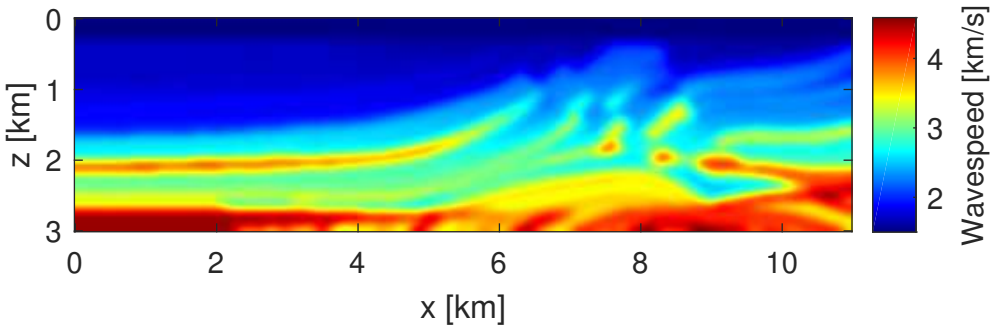
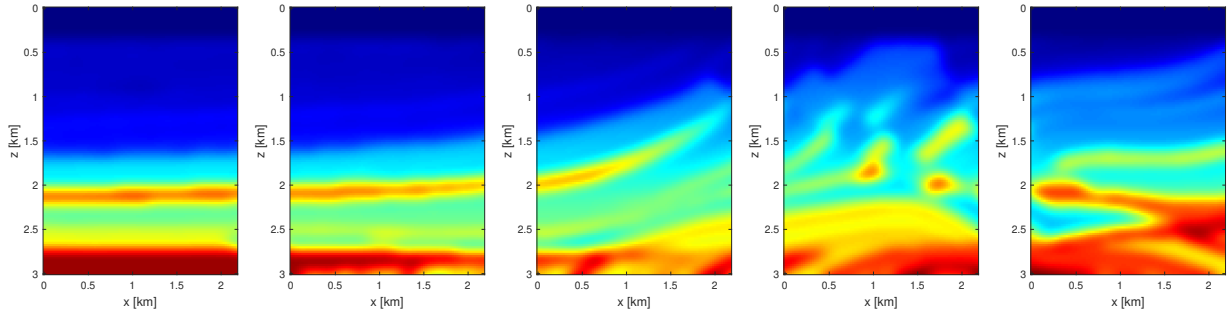


Figure 6.2.1: *Smooth Marmousi model used for Experiment 2.*

Experiment Setup: We discretise this model into a 440×121 grid, with a grid spacing of 25 m in both the x and z direction. We split this model horizontally into five slices of equal size, with the aim to use four of these slices as training models, on which our algorithm will learn the optimal parameters, and one as a testing model, on which the performace of the optimal parameters will be evaluated. This idea has been motivated by experiments performed in [85] and [84]. The Marmousi slices are presented in Figure 6.2.2. For our experiment, we choose Slice 1 as the testing model, and Slices 2 to 5 as the training models.



(a) Slice 1 (b) Slice 2 (c) Slice 3 (d) Slice 4 (e) Slice 5

Figure 6.2.2: *Smooth Marmousi model divided into individual slices.*

6.2.1 Training

Experiment Details: For clarity, we display the training models only in Figure 6.2.6, and point out here that the corresponding data does not involve added noise. We aim to optimise the positions of ten sensors, and the value of the Tikhonov regularisation

parameter. The starting acquisition setup is overlaid on Slice 2 in Figure 6.2.3, and this setup is equivalent for every slice. Ten sources are positioned uniformly along a line on the left hand side of the domain, and the ten sensors are uniformly spaced along a line on the opposite side of the domain. Therefore, our initial guess at the sensor positions produces a transmission acquisition setup. Our initial guess at the Tikhonov regularisation parameter is 2.5, chosen through experimentation because it was observed to generate reasonable reconstructions for the training models in general. We make these initial guesses at sensor positions and Tikhonov regularisation parameter as we want to start from a setup that could be typical of a geophysical problem. The model starting guess for FWI is a smooth vertical variation of wavespeed, increasing with depth, and containing no information on any of the structures in the training models. The FWI starting guess for each training model is shown in Figure 6.2.4.

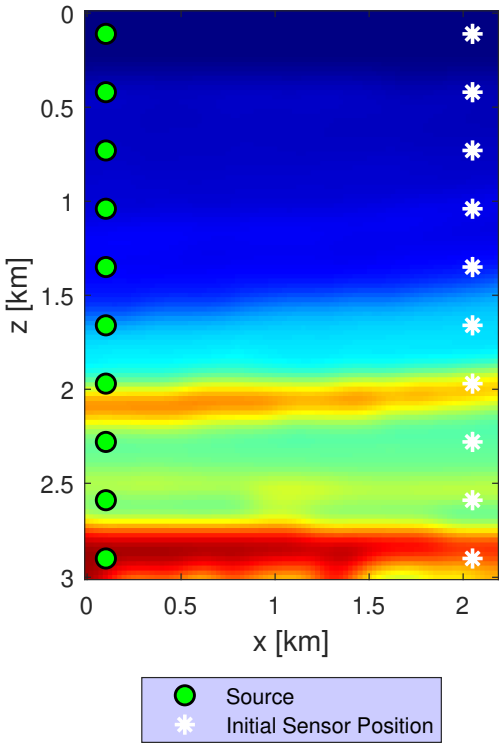


Figure 6.2.3: *Initial sensor positions for Experiment 2. The sensors are uniformly spaced along a vertcial line such that this is a transmission acquisition setup. This setup is shown on Slice 2 but the setup is the same for each training model.*

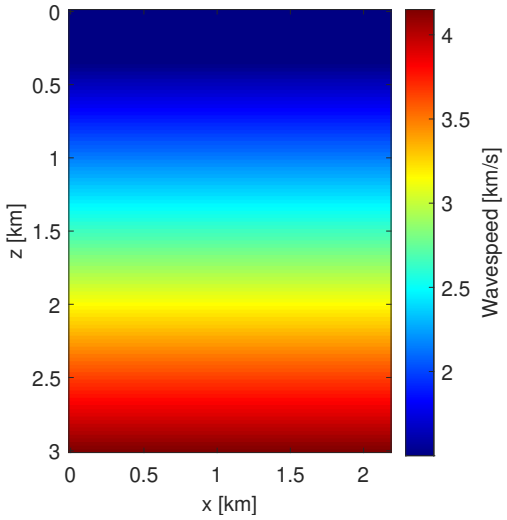


Figure 6.2.4: *Initial wavespeed guess for FWI for each training model. This initial model consists of a one-dimensional variation of wavespeed with depth.*

We apply our bilevel frequency continuation approach using four frequency groups in the range 0.5 Hz to 6 Hz. In the first three groups, the sensor positions are optimised and in the final frequency group the regularisation parameter is incorporated. The stopping conditions used are the same as Experiment 1. That is, the lower-level problem is solved to a tolerance of $||\nabla\phi||_2 \leq 10^{-10}$, the linear system involved in the upper-level gradient computation is solved to a tolerance of 10^{-15} and each frequency group of the upper-level is iterated until the infinity norm of the projected gradient is smaller than 10^{-10}, the updates to ψ or the optimisation parameters stall or the maximum number of iterations is reached (50 iterations).

Results: The final positions converged to are displayed on Slice 2 in Figure 6.2.5. We see that many sensors have remained on the right-hand side of the domain, close to their starting guesses, but are no longer uniformly spaced along a straight line. The sensors on the bottom half of the domain are more widely spaced than the sensors on the top half of the domain. These sensors on the right record the wavefield which has been transmitted from the sensors through the domain. We also observe that some sensors have converged to positions near the top of the domain, again in the right half of the domain. We expect that these sensors measure both waves transmitted through the domain, and waves which have been generated by the sources near the top of the domain and reflected from the layers of higher wavespeed. The multiple sensors near the top of the domain are then, in a way, a combination of transmission and reflection setups. A plausible reason for why there are more sensors in the top half than the bottom half of the domain is that there are more reflections occurring in this direction (due to the general increase in wavespeed with depth). Finally, there is one sensor further into the interior of the domain, which is seen in Figure 6.2.5 to lie just above the layer of higher wavespeed in Slice 2. We note that for Slice 5, this sensor position also

lies just above a higher wavespeed layer, and in Slice 4 this sensor is placed just under the upward sloping higher wavespeed layer. The optimal regularisation parameter for this setup is found to be 6.77×10^{-3}.

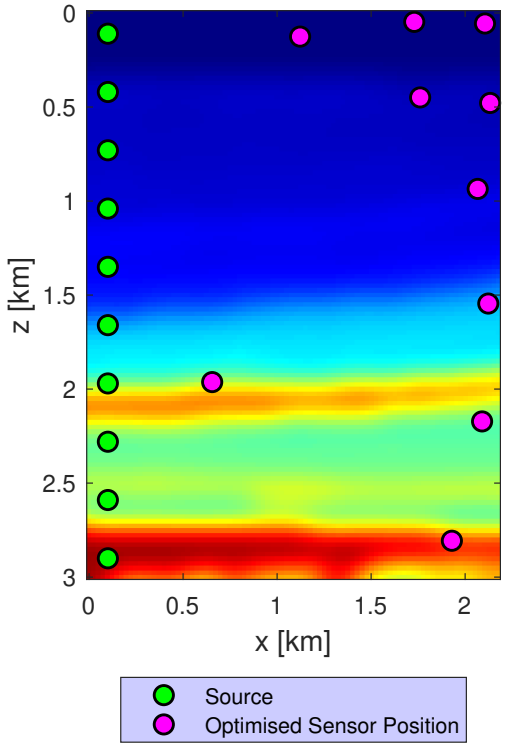
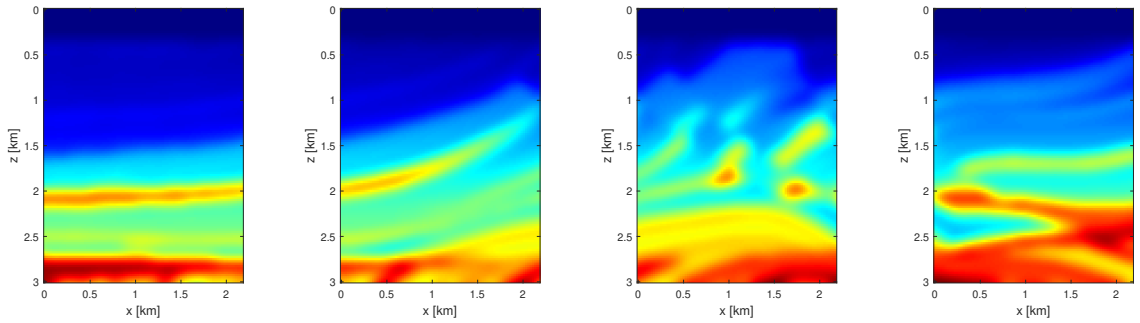


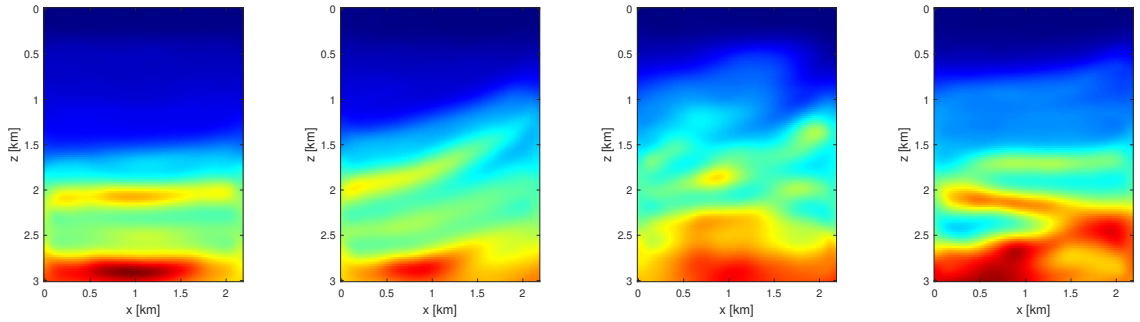
Figure 6.2.5: *Optimised sensor positions for Experiment 2 displayed on Slice 2.*

For comparison, we display the ground truth training images in Figure 6.2.6, the corresponding reconstructions at the initial guess (i.e., for the setup in Figure 6.2.3) in Figure 6.2.7, and the reconstructions at the optimal parameters (i.e., for the setup in Figure 6.2.5) in Figure 6.2.8. We observe that the starting parameters produce reconstructions that, in general, identify the large-scale structures present in the ground truth. However, the shapes and wavespeed values of the structures are not always correct. For example, in Slice 2, the upper-layer of higher wavespeed at around 2.1 km in depth, has not been reconstructed at the edges of the domain, and the values of the wavespeed in the layer are too low. In Slice 3, the upward sloping layer of higher wavespeed, beginning at a depth of 2 km, is also not reconstructed at a high enough wavespeed, and is barely noticeable against the surrounding wavespeed. In comparison to the other slices, the reconstruction of Slice 4 is relatively poor as many of the details present in the ground truth are missing. For Slice 5, the general structure of the ground truth is present, but the wavespeed values are incorrect and some finer details are missing. In the reconstructions at the optimised parameters (Figure 6.2.8), many of the issues listed have been improved on. Both the geometry and wavespeed values in these reconstructions appear closer to the ground truth. The images have appeared to 'sharpen' up, particularly in the upper part of the domain, and the finer features of structures and boundaries between the layers have become evident.



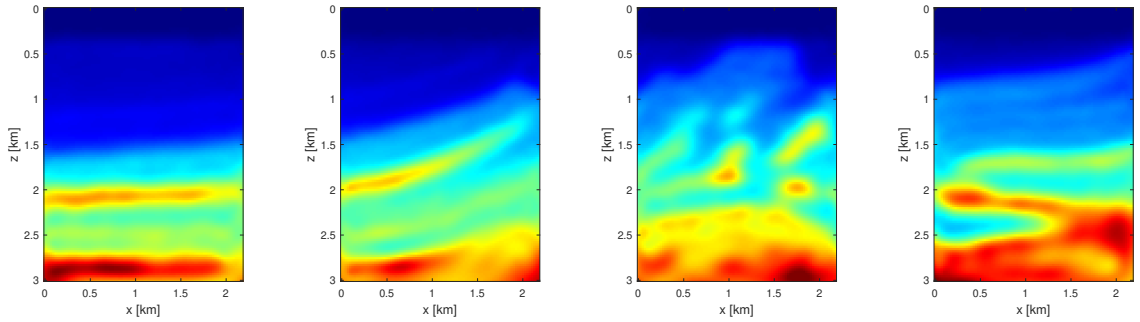
(a) Slice 2 (b) Slice 3 (c) Slice 4 (d) Slice 5

Figure 6.2.6: *Ground truth training models used in the bilevel algorithm to learn the optimal sensor positions and the Tikhonov regularisation parameter.*



(a) Slice 2 (b) Slice 3 (c) Slice 4 (d) Slice 5

Figure 6.2.7: *Reconstructions of training models at the initial sensor positions and Tikhonov regularisation parameter.*



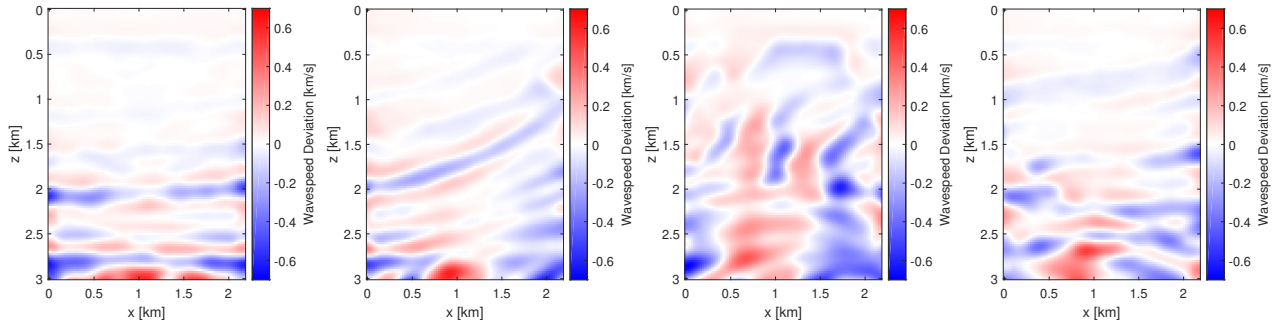
(a) Slice 2 (b) Slice 3 (c) Slice 4 (d) Slice 5

Figure 6.2.8: *Reconstructions of training models at the optimised sensor positions and Tikhonov regularisation parameter.*

Further comparison between the reconstruction at the starting guess and the reconstructions at the optimal parameters is found through the use of improvement factors, which are reported for each slice in Table 6.2.1. The improvement factors range from 4.49 and 12.23 (rounded to two decimal places), showing a consistent improvement across training models. Although these improvement factors are lower than those seen

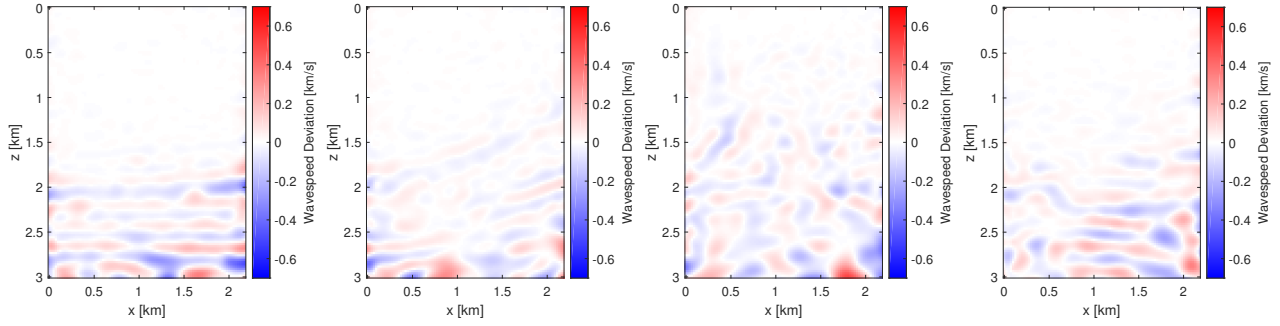
in Experiment 1 in Section 6.1, this level of improvement is still significant for a realistic geophysical problem.

For more insight into the quality of the FWI reconstructions of the training models, we visualise the errors in both the starting and optimised reconstructions. Figures 6.2.9 and 6.2.10 display the deviation in the reconstructions from the ground truth in terms of wavespeed values (i.e., we have plotted the error defined by (2.6.1)). The blue indicates when the reconstructed wavespeed value is lower than the ground truth value, and the red indicates when the reconstructed wavespeed value is higher than the ground truth value. The darker the colour, the further the reconstruction is from the ground truth, with white indicating zero error. These plots make the difference in the starting and optimised reconstructions clear. In the starting reconstructions, the deviation from the ground truth is noticeable throughout each slice. In the optimised reconstructions, this error is greatly reduced, particularly in the upper part of the domain, where the error is small or close to zero. Errors at the bottom of the domain are still noticeable, particularly at a depth over 2km, which we expect is related to the fact that there are less sensors in this part of the domain.



(a) Slice 2 (b) Slice 3 (c) Slice 4 (d) Slice 5

Figure 6.2.9: *Deviation in the initial FWI reconstructions from the ground truth training models.*

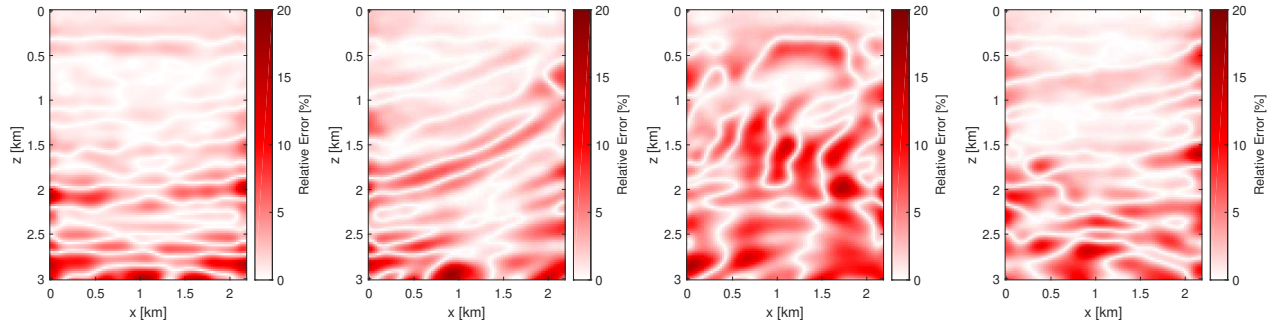


(a) Slice 2 (b) Slice 3 (c) Slice 4 (d) Slice 5

Figure 6.2.10: *Deviation in the optimised FWI reconstructions from the ground truth training models.*

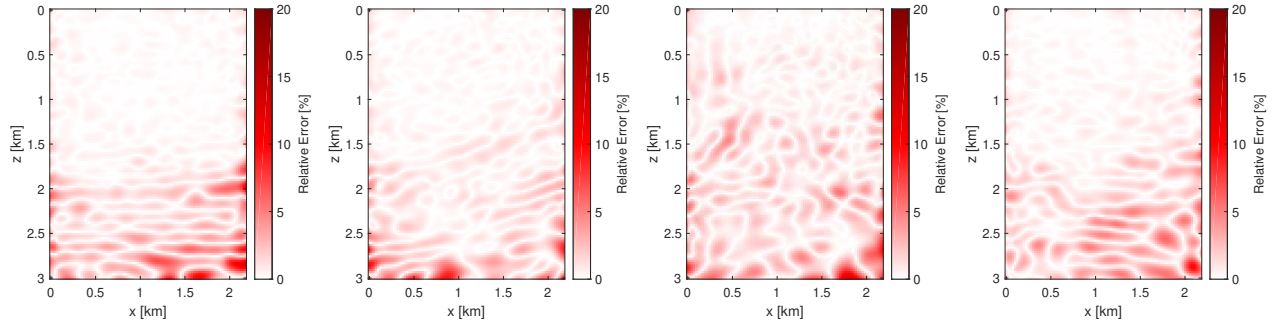
The relative percentage errors (computed by (2.6.2)) for the starting and optimised reconstructions are displayed in Figures 6.2.11 and 6.2.12 respectively. We report the

mean relative percentage error for the optimised reconstructions in Table 6.2.1. Table 6.2.1 shows that, on average, the optimal parameters result in errors that are low for all training models, ranging from 0.746% to 1.057%. Using this metric of relative percentage errors, Slice 3 has the best reconstruction and Slice 4 has the worst (but Slice 4 does have the best improvement factor). We note here that Slice 2 has the best starting guess in terms of this metric (and in term so ψ), which partly explains why its improvement factor is the lowest.



(a) Slice 2 (b) Slice 3 (c) Slice 4 (d) Slice 5

Figure 6.2.11: *Absolute value of the relative percentage error in the reconstructions at the initial parameters.*



(a) Slice 2 (b) Slice 3 (c) Slice 4 (d) Slice 5

Figure 6.2.12: *Absolute value of the relative percentage error in the reconstructions at the optimised parameters.*

In Table 6.2.1, we also report the structural similarity between the ground truths and the optimised reconstructions using a measure called the Structural Similarity Index (SSIM). The SSIM is a quality metric commonly used in imaging [182]. A good similarity between images is indicated by values of SSIM that are close to 1. We can see from Table 6.2.1. that all values of the SSIM measure for the optimised reconstructions are over 0.95, which implies the optimised reconstructions are structurally very close to the ground truths. We note that the SSIM values here have been computed using the `ssim` function in Matlab's Image Processing Toolbox.

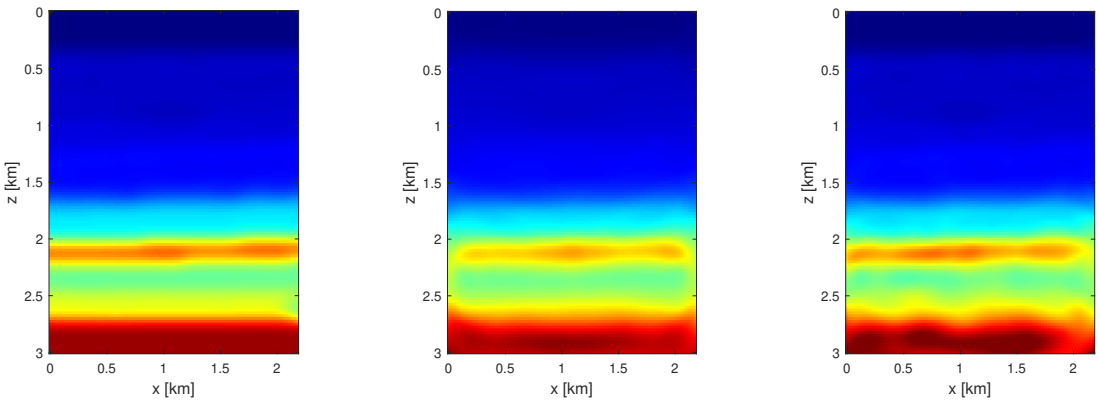
Slice	Improvement Factor	Mean Relative Error (%)	SSIM
2	4.49	0.920	0.960
3	8.26	0.746	0.969
4	12.23	1.057	0.960
5	6.65	0.903	0.959

Table 6.2.1: *Measures of improvement/quality of the FWI reconstructions of training models at optimised parameters.*

6.2.2 Testing

We evaluate the optimal parameters found by our bilevel algorithm by testing them on Slice 1 of Figure 6.2.2.

Results: We compare the FWI reconstructions of the testing slice at the starting setup in Figure 6.2.3, i.e., with the uniform placement of sensors along a vertical line, and the optimised setup found by our algorithm in Figure 6.2.3. The ground truth, reconstruction at the initial parameters, and reconstruction at the optimised parameters are shown in Figure 6.2.13.

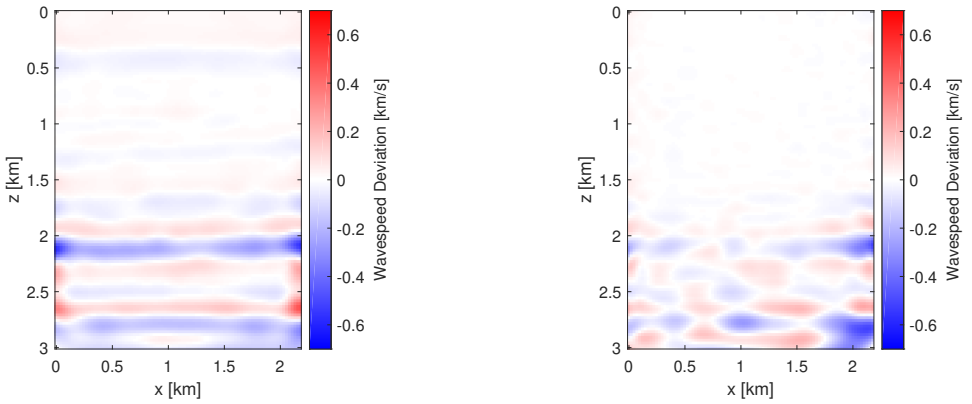


(a) Ground Truth (b) Initial (c) Optimised

Figure 6.2.13: *Ground truth testing model and its correspondng reconstructions at the initial and optimised paramters.*

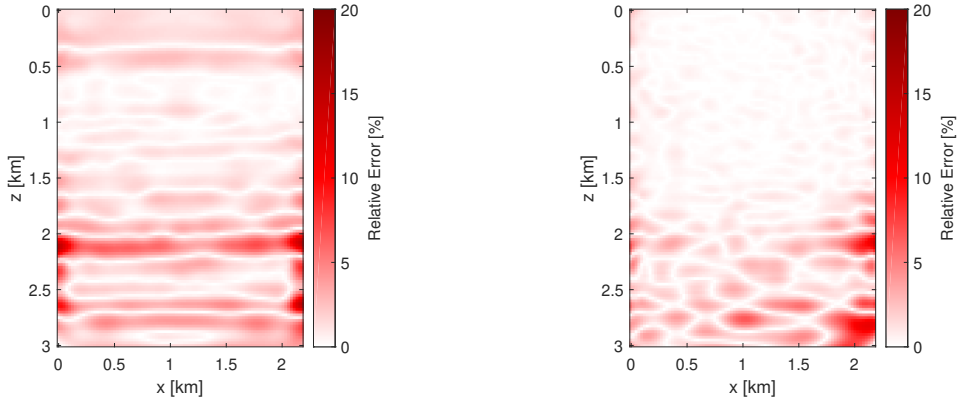
We observe that the reconstruction at initial parameters appears to be of relatively good quality already, but that the values of the wavespeed, in the thin layer below 2 km in particular, seem to be incorrect. The optimised parameters appear to have improved on this. We compare the initial and optimised reconstructions in more detail by visualising their wavespeed deviation and relative percentage error in Figures 6.2.14 and 6.2.15 respectively. It is evident from these figures that the optimal parameters determined by our algorithm yield a substantial improvement in the FWI reconstruction. Similarly to the training models, the error becomes lowest in the part of the domain

above 2km, and errors are more noticeable below this, particularly on the right side of the domain. There is relatively large error in the bottom right corner, which can possibly be explained by the position of the sensor in this corner. This sensor is positioned at 1.92 km in the x direction, and so is not measuring the waves being transmitted though the region 'behind' it, and therefore is not reconstructing this region correctly. In general, however, many of the errors present in the initial reconstruction have been improved on.



(a) Initial (b) Optimised

Figure 6.2.14: *Deviation in the initial and optimised FWI reconstructions from the ground truth testing model.*



(a) Initial (b) Optimised

Figure 6.2.15: *Absolute value of the relative percentage error in the FWI reconstructions of the testing model at the initial and optimsied parameters.*

The values of improvement factor, mean relative error and SSIM are reported in Table 6.2.2. From these values we can conclude that the reconstruction at the optimal parameters is indeed of good quality, and that there is a significant improvement from

the starting reconstruction. Moreover, the reconstruction of the testing model is of similar quality to the models used for training.

Slice	Improvement Factor	Mean Relative Error (%)	SSIM
1	4.46	0.910	0.969

Table 6.2.2: *Measures of improvement/quality of the FWI reconstruction of the testing model at optimised parameters.*

Conclusion: We note that the testing model (Slice 1) and one of the training models (Slice 2) appear to be in the same class of models, which explains why their improvement factors are close in value. The other training models are not in the same class as the testing model, or as each other, but share some properties, such as the range of wavespeeds present and the fact that the wavespeed gets higher on average as depth is increased. This is in contrast to Experiment 1 where all training models were in the same class. The results of this Experiment 2 show that we still achieve consistent improvement when training on a diverse set of models, and that we achieve desirable results when testing on a model that is in the same class as only one of the training models.

Overall, this experiment has shown that our bilevel algorithm is effective in finding the optimal sensor positions and Tikhonov regularisation parameter for geophysical problems, and that the optimal parameters can be applied successfully to subsurfaces that are overall largely different from the training set, as long as there are some similarities present.

Chapter 7

Discussion

In this thesis, we studied the problem of optimising the quality of seismic images produced with full waveform inversion (FWI). We formulated this problem using a bilevel learning framework, and developed and implemented an iterative solution approach. We showed that, using this supervised machine learning approach, both the sensor positions and regularisation parameters can be optimised for FWI, and that our method leads to large improvements in the quality of FWI images. We summarise some of the main results of this thesis here and discuss the potential for future work.

7.1 Summary of Results

In Chapter 2, we provide an overview of the FWI method. This involves a discussion of the forward modelling step, a derivation of the gradient and Hessian of the FWI objective function (including the application of the adjoint-state method), a description of the FWI algorithm, and some examples of FWI reconstructions. The original work in this chapter focuses on the FWI Hessian. We provide an in-depth analysis into the structure and properties of the Hessian, and derive lower- and upper-bounds on the eigenvalues of the Hessian. We provide conditions for a positive-definite Hessian, in terms of a bound on the regularisation term of the convex parameter, and hence make conclusions about when the FWI problem has a unique solution. This result is important for both the standalone FWI problem, and also the overall parameter optimisation problem.

The work in Chapter 3 solely consists of all original results. The parameter optimisation problem, for the optimisation of both sensor positions and the Tikhonov regularisation parameter, is formulated as a bilevel learning problem. A solution approach to the bilevel problem is proposed, involving a single-level reduction and a gradient-based local optimisation method. Novel formulae for the gradient of the bilevel problem are derived, incorporating the application of the adjoint-state method. The cost of solving the bilevel problem in terms of the number of PDE solves is studied. Finally, the smoothness of the upper-level objective function is proved, showing the applicability of a gradient-based optimisation method.

The main result of Chapter 4 is that, under certain assumptions, the solutions to

both the lower- and upper-level problems have symmetry properties.

In Chapter 5, the algorithms used to solve the bilevel problem are presented, which include several original algorithms. Implementation details and novel ideas for the improvement in the performance of these algorithms are described. Specifically, we propose a bilevel frequency continuation algorithm, which is demonstrated to significantly improve the performance of the bilevel algorithm. Using this continuation scheme, the parameters being optimised have the potential to move far from their starting guess without getting stuck in local minima. In addition, two new preconditioners are devised and implemented to reduce the number of iterations taken to solve a linear system involving the FWI Hessian. We observe that both preconditioning strategies work effectively to speed up the solution of the linear system, producing a reduction in the number of iterations of up to 96%. We present our investigation of how each preconditioning strategy is affected by various parameters, hence demonstrating that both preconditioners are unaffected by grid size. We show that when used in the bilevel algorithm, both preconditioning strategies improve the performance of the bilevel algorithm significantly, reducing the computational time by several hours in some cases. We note that these preconditioning techniques have the potential to be useful in solving any Hessian system, not just the one that appears in the bilevel problem. As part of Chapter 5, we also provide an overview of the computational time spent on different parts of the algorithm, and demonstrate that our new bilevel algorithm parallelises effectively over the number of training models.

While each chapter demonstrates aspects of the numerical implementation of the bilevel problem, the experiments in Chapter 6 involve applying the algorithm to larger-scale problems. The experiments in this chapter show that the bilevel learning approach to parameter optimisation performs well on larger-scale problems, including realistic geophysical problems, and that the optimal parameters generalise well to models that are outside the training set, even if there are some significant differences between the model and the training set. This makes our bilevel learning algorithm extremely valuable, since, even though it is computationally expensive to run, it only needs to be trained once to determine optimal parameters that can then can be applied with some confidence to a relatively diverse range of situations.

7.2 Future Work

There are many possibilities for extending, improving, and applying the work done in this thesis. We consider various potential areas for development here.

Other Types of Imaging Procedures Firstly, the work done in this thesis provides a framework for which other types of images, outside seismic images, can be improved. In particular, our work naturally extends to other applications that use FWI as a reconstruction technique, for example ultrasound imaging ([81], [120]), where our work can be used to optimise both sensor positions and regularisation parameters for the improvement in the quality of the medical images, and structural health monitoring ([159], [188]), where our method can be used to improve the accuracy of non-destructive

testing. Our bilevel learning theory and algorithms also have the potential to be developed and applied to other imaging techniques where the forward modelling, and hence the lower-level, is different, but where the general bilevel approach that we have developed would still be useful. Medical imaging applications, either modelled as linear or nonlinear inverse problems, such as electrical impedance tomography (EIT), X-ray tomography, and computed tomography (CT), could all benefit from our parameter optimisation techniques.

Optimising the Number of Sensors Another suggestion for extending the work in this thesis is to incorporate the optimisation of the number of sensors into the bilevel learning structure. This can be formulated by including a weighting operator applied to the restriction operator in the FWI objective function, such that each sensor has a weight between zero and one. Then both the positions and the weight of each sensor can be optimised on the upper-level. A sparsity penalty term for the weights, such as the one used in [162], can be included in the upper-level objective function to encourage the weights to take either the value zero or one, depending on how important the sensor is to the quality of the reconstruction.

Optimising the Source Positions The bilevel problem can be enhanced even further to include the optimisation of source positions (as well as the number of sources) so that the whole acquisition set up is optimised. The extension for source placement optimisation would involve the same steps as the sensor case, namely: deriving formulae for the gradient of the upper-level objective function with respect to source positions, proving smoothness of the upper-level objective function with respect to source position, and incorporating the optimisation of this into the existing algorithm. There is also potential to study source-receiver reciprocity (as described in [104]) and to determine whether this can be incorporated into the algorithm to somehow improve efficiency.

Further Numerical Experiments Another area for future development is related to numerical experiments. The numerical experiments in this thesis are generally set up as transmission/crosswell type experiments. When optimising the sensor positions in these experiments, the sensors were usually allowed to move anywhere in the domain, to reach their optimal positions. This resulted in large improvements in the quality of the FWI images. Further experiments can be done to examine how well the bilevel learning approach works for reflection setups and for any other setups where there are bounds on where the sensors can be placed. These types of experiments may be closer to the settings encountered in real seismic acquisition, where more constraints are present. For example, there is usually a preference for placing the sensors on the surface (i.e., a reflection setup), due to the expense of drilling boreholes. In addition, even if a transmission set up is chosen, it is not expected that boreholes can be drilled in several locations, and in reality there are practical constraints on where these can be drilled. The current implementation of the bilevel algorithm already supports this kind of extension - only the inputs to the algorithm need to be changed, for example the starting guess for the sensor positions and the bounds for the L-BFGSb algorithm.

Open Questions Arising from Chapter 2 There are a number of theoretical open questions that have resulted from this research. In Chapter 2, we found a bound on the convex regularisation parameter that would ensure a unique FWI solution. Since there are several unknown constants in this bound, a question that arises from this result is how large this parameter actually needs to be to ensure uniqueness. Further investigation can be taken to estimate these unknown constants, along with analysis into how the other terms in the Hessian can help with establishing its positive-definiteness. As mentioned in Remark 2.4.19, it is likely that we have over-estimated how large the convex regularisation parameter needs to be to achieve positive-definiteness in the Hessian. This seems to be confirmed by what we have observed in practice. The bound we have derived could therefore be refined further.

Open Questions Arising from Chapter 3 Several open questions also arise from the work presented in Chapter 3. In Chapter 3, we proved that the upper-level objective function is continuously differentiable with respect to sensor position. An interesting extension of this work would be to prove that the objective function is twice continuously differentiable and to derive an expression for the upper-level Hessian. This work would be useful in the context of investigating upper-level uniqueness. In Chapter 3, we briefly examined the possibility of a unique upper-level solution. This was considered in a simplified case with one sensor. Having shown that a unique solution is possible in the one sensor case, this work can be continued to examine the multiple sensor problem. In this case, it would be useful to derive a bound on a possible upper-level regularisation parameter such that a unique solution is guaranteed (much like the approach taken on the lower-level). Another topic from Chapter 3 that should be explored in depth is the difference between the noisy data and clean data case. In the Problem Formulation section (Section 3.3), we proposed adding noise to the synthetic 'observed' data. Later, in proving the smoothness of the upper-level objective function we assumed that there was no noise on the data. It would be of interest to investigate whether the upper-level objective function remains smooth when the data is noisy. This is not a straightforward problem, since in practice it was observed that the objective function was only smooth when there was no noise in the synthetic data.

Open Questions Arising from Chapter 4 An immediate extension of the work in Chapter 4 is to generalise the results on reflections to general rotations.

Alternative Approaches to Solving the Bilevel Learning Problem Finally, it is worthwhile to consider different approaches to solving the bilevel learning problem, other than just a local optimisation method that uses gradient information to find the minimum. As we have noted, noisy data results in a non-smooth objective function. Similarly, we showed that the objective function becomes non-smooth when the lower-level is not solved accurately enough (see Figure 5.3.5). However, the overall shape of the objective function (including its global minimum) remains about the same. Therefore, non-smooth optimisation methods could be applied to solve these problems. This would mean that FWI would not have to be solved as accurately to solve the full bilevel

problem, and hence a large amount of computation time could be saved (since, as noted in Section 5.3.3, most of the computation time is spent solving the FWI problem). There are many derivative-free optimisation methods, as detailed in [49], which may be be useful for our problem. In particular, surrogate modelling, which involves replacing the true objective function with an approximation to it that is cheaper to evaluate, could be a useful approach. Although the surrogate function is less accurate than the true objective function, several evaluations of a surrogate model can still be less expensive than one evaluation of the true function and its gradient [49].

In conclusion, there are many possible research directions that can build on the research done in this thesis.

Appendix A

Full Waveform Inversion Background

Full Waveform Inversion started to emerge in the 1980's, with pioneering publications by Lailly [111] and Tarantola [175,176]. Since then, application to real seismic exploration data has become common due to advances in seismic-data acquisition and improvements in computer capabilities.

Lailly and Tarantola laid out a detailed formulation for the recovery of subsurface parameters using a least squares optimisation scheme, which involved using the adjoint-state method for the calculation of the descent direction. The first computer implementations of this theory were published by Gauthier et al. [75] in 1986. Gauthier et al. implemented the inversion algorithm on various acoustic 2D examples to test it's performance. Despite the computational limitations at the time, this work proved the feasibility of the method, and so represented a big advancement in the field.

Although FWI was originally posed in the time domain (for example, [175]), it was later formulated and applied in the frequency domain, for example, by Pratt and Worthington [151], and Pratt ([148] and [149]).

One of the main issues that FWI faced in its application to real seismic data was that, if the initial model is not sufficiently accurate, FWI often converges to incorrect results. This problem is due to the numerous local minima of the objective function, and is associated with using gradient-based descent methods to solve FWI. Convergence to a local minimum produces erroneous results and an improper interpretation. Significant progress in this challenge were made with the proposal of hierarchical strategies, both in the time domain ([38]) and frequency domain (for example, [147], [150], [149]). This strategy in the frequency domain (frequency continuation) is further discussed in Section 2.5.1, and we summarise the main concept here. The key idea is to start from low frequency data, which corresponds to long wavelengths. Waves interact with with features on the order of their wavelength, and so, following some FWI iterations, the inverted model will contain the large-scale structure. Higher frequencies are then progressively introduced and the model is refined accordingly. This strategy helped to avoid local minima and hence improved FWI results. Sirgue and Pratt [166] further improved the process by showing that carefully choosing the frequencies can speed up the FWI process.

Since then, the advancements in High Performance Computing have meant that FWI could be applied successfully to large scale and realistic problems. Several case studies have demonstrated impressive FWI results, for both marine data [145,165] and

land data [144]. Despite being widely used for seismic surveys in industry today, FWI still has some issues. For example, when there is a lack of low frequencies or when the data is noisy, FWI can produce large artefacts. In addition, even with the advancements in computing, FWI is still extremely computationally intensive. It should also be noted that FWI is an ill-posed problem inverse problem, with many solutions that may not make any geological sense, an issue which is related to the large number of model parameters.

Advancements are continuously being made in the area of FWI. Recent work includes improving the robustness of FWI by suppressing artefacts [189], as well as frequency-dependant preconditioning that improves the reconstruction of deep geological structures [55]. It is also currently becoming more popular to apply machine learning techniques to improve FWI reconstructions, seen for example in [192] and [173]. New applications of FWI have also been researched recently, such as [81], which demonstrates that FWI can be used to image the brain accurately, and [120], which uses FWI to create 3D ultrasound images that can replace X-ray-based mammography.

Appendix B

Seismic Waves

Seismic waves are mechanical waves that propagate through the Earth. The source of seismic waves may be natural or man-made. An earthquake is an example of a natural source, where the seismic waves are generated at a 'hypocentre', deep in the subsurface, along a geological fault, and these waves spread out through the Earth, possibly reaching the surface. Man-made seismic waves are due to an artificially generated explosion, and the measurement of these seismic waves can be used in applications such as hydrocarbon exploration.

There are two main categories of seismic waves - *body waves*, which propagate within the interior of the Earth, and *surface waves*, which propagate across the surface. Each of these categories can be further divided into sub-categories. We present a brief description of these here. For a more detailed description, see [137, Chapter 1].

Body waves are separated into the following classes: P-waves, also called *primary* or *compressional* waves, and S-waves, or *secondary* or *shear* waves. P-waves are pressure waves and propagate longitudinally in the direction of the path of propagation, therefore dilating and compressing the medium. P-waves travel faster than other waves and can travel through any type of material (solid, liquid, gas). P-waves are modelled by the *acoustic wave equation*. The S-waves are shear waves that displace the ground perpendicular, or transverse, to the direction of propagation. S-waves only travel through solid mediums, as shear force does not occur in liquids and gases. Both P- and S-waves can be modelled by an *elastic wave equation*.

Surface waves travel slower than body waves, but their particle movement is much more pronounced, and so during earthquakes, they cause the most damage. Surface waves can also be separated into different classes. *Love* waves, named after Augustus Edward Hough Love, are the fastest of the surface waves. Love waves involve surface motion which is perpendicular to the direction of propagation. *Rayleigh* waves are named after John William Strutt Rayleigh. Rayleigh waves involve the elliptical movement of particles, against the propagation direction, resulting in motion of a rolling nature, similar to ocean surface waves.

Appendix C

Formulations of the Wave Equations

From a mathematical point of view, wave propagation is governed by the so-called *wave equations*. The main objective of this section is to introduce the elastic wave equation and derive the acoustic wave equation from this.

Elastic Waves

A sophisticated description of seismic waves must take into account that the Earth is a solid, and hence elastic phenomena are important. Elastic waves in solid materials can be modelled using the equations of linear elastodynamics. These equations are derived from classical (Newtonian) mechanics. We present the elastic wave equation here without derivation, however, the derivation and more details can be found in [167].

The isotropic elastic wave equation may be written as

$$\rho(x)\frac{\partial^2 \mathbf{u}(x,t)}{\partial t^2} - \nabla\cdot(\lambda(x)\nabla\cdot\mathbf{u}(x,t)) - \nabla\cdot\left(\mu(x)\left[\nabla\mathbf{u}(x,t) + (\nabla\mathbf{u}(x,t))^T\right]\right) = 0, \tag{C-1}$$

where t is the time variable and x is the spatial variable (which can represent more than one dimension). The wavefield here is the displacement, denoted by $\mathbf{u}$. We use bold notation for the displacement to emphasise that it is a vector, with size given by the number of dimensions. The density of the medium is denoted by ρ, and the two elastic parameters, or Lamé parameters, are denoted by λ and μ. The parameter λ is referred to as the first Lamé parameter, and μ is referred to as the second Lamé parameter, or *shear modulus*. Other physical coefficients can be expressed with respect to the Lamé parameters, such as the bulk modulus,

$$\kappa(x) = \lambda(x) + \frac{2}{3}\mu(x). \tag{C-2}$$

In a homogeneous medium, there is no spatial variations in the density and Lamé parameters. In this case (C-1) can be written as

$$\rho\frac{\partial^2 \mathbf{u}(x,t)}{\partial t^2} = (\lambda+2\mu)\nabla(\nabla\cdot\mathbf{u}) - \mu\nabla\times(\nabla\times\mathbf{u}). \tag{C-3}$$

Remark C-1. ***P- and S-Wave Decomposition:*** *As previously stated in Appendix B, elastic body waves propagating through a solid are made up of both P-waves and S-waves. We can see this mathematically by separating the elastic wave equation (C-3) into solutions for P- and S-waves using the Helmholtz decomposition. The Helmholtz decomposition involves expressing the displacement vector* $\boldsymbol{u}$ *as a composition of two functions,*

$$\boldsymbol{u} = \boldsymbol{u}^{(s)} + \boldsymbol{u}^{(p)},$$

where $\boldsymbol{u}^{(p)}$ *corresponds to P-waves (compressive waves) and* $\boldsymbol{u}^{(s)}$ *corresponds to S-waves (shear waves). The compressive waves* $\boldsymbol{u}^{(p)}$ *satisfy* $\nabla \times \boldsymbol{u}^{(p)} = 0$ *and shear waves* $\boldsymbol{u}^{(s)}$ *satisfy* $\nabla \cdot \boldsymbol{u}^{(s)} = 0$*. For homogeneous materials, each of these is the solution to the following separate wave-equations*

$$\left(\frac{\partial^2}{\partial t^2} - c_p^2 \Delta\right) \boldsymbol{u}^{(p)} = 0 \tag{C-4}$$

$$\left(\frac{\partial^2}{\partial t^2} - c_s^2 \Delta\right) \boldsymbol{u}^{(s)} = 0 \tag{C-5}$$

where c_p *and* c_p *are given respectively by*

$$c_p = \sqrt{\frac{\lambda + 2\mu}{\rho}} \tag{C-6}$$

$$c_s = \sqrt{\frac{\mu}{\rho}}. \tag{C-7}$$

Acoustic Approximation

The elastic wave equation (C-1) models the propagation of both P and S waves in an elastic solid. Modelling a full elastic wavefield is computationally expensive, however, and simplifications can be introduced to help. Since P-waves travel faster than S-waves, and so bring information first, some applications require P-waves only. In addition, only P-waves propagate in fluids, which cannot support shear waves, as fluids have no aspect of rigidity. Therefore, a common simplification of the elastic wave equation is the *acoustic approximation*, which models P waves only. The following derivation of the acoustic approximation are based on the steps outlined in [72, Section 1.2.6].

Under the acoustic approximation, there is no shear waves and therefore the shear wave velocity and shear modulus are zero ($c_s = 0$, $\mu = 0$). This means that terms that represent the deformations that give rise to shear stresses will vanish. Therefore, Equation (C-1) is simplified to

$$\rho(x) \frac{\partial^2 \mathbf{u}(x,t)}{\partial t^2} = \nabla \left(\lambda(x) \nabla \cdot \mathbf{u}(x,t)\right), \tag{C-8}$$

which is the acoustic isotropic wave equation. The equation for the P-wave velocity (C-6), becomes

$$c_p(x) = \sqrt{\frac{\lambda(x)}{\rho(x)}}. \tag{C-9}$$

By (C-2), when the shear modulus is zero, the first Lamé parameter is the bulk modulus, and so $\lambda(x) = \kappa(x) = c_p^2(x)\rho(x)$.

The elastic equation (C-1) requires the computation of the displacement $\mathbf{u}$, which is a vector. An important simplification of the acoustic case is that it is possible to introduce the pressure $p = p(x,t)$, a scalar quantity, as the unknown instead. When shear is zero, the waves can be described by Cauchy's momentum equation for a non-viscous fluid. This provides us with a link between displacement and pressure, from which we obtain a simple expression for the acoustic pressure,

$$p(x,t) = -\lambda(x)\ \nabla \cdot \mathbf{u}(x,t). \tag{C-10}$$

Using (C-10) to substitute for $\mathbf{u}$ in (C-8), and dividing (C-8) by ρ gives the following,

$$\frac{\partial^2 \mathbf{u}(x,t)}{\partial t^2} = -\frac{\nabla p(x,t)}{\rho(x)}.$$

Taking the divergence of this, and again using the definition (C-10), gives the following equation,

$$\frac{\partial^2 p(x,t)}{\partial t^2} = \lambda(x)\nabla \cdot \left(\frac{\nabla p(x,t)}{\rho(x)}\right).$$

This wave equation is the acoustic approximation to elastic waves.

Further *assuming that density ρ is constant*, and multiplying the above by the constant ρ, gives

$$\frac{1}{c_p^2(x)}\frac{\partial^2 p}{\partial t^2} = \nabla^2 p, \tag{C-11}$$

where $c_p(x)$ is defined in (C-9). This is the scalar acoustic wave equation (with constant density), without a source term (body force term). A derivation with a source term is found in [72, Section 1.2.6]. The acoustic approximation (C-11) is widely used in geophysical applications. It is of importance in the context of inverse problems because there are less parameters to identify in the model. The acoustic wave equation is also the natural equation to model sounds waves in fluids, and so is applied to model sound waves in the water during off-shore marine acquisition.

Appendix D

Wave Equation Formulation in the Frequency Domain

The wave equations (C-1) and (C-11) have been formulated in the time domain. The unknowns, $\mathbf{u}$ and p, depend on the time-variable t and the spatial variable x. In this thesis, we will deal with the wave equations in the *frequency domain*, also known as the time-harmonic formulation (e.g., see [48]). In this formulation, the unknowns depend on the spatial variable x and the angular frequency ω.

The time-harmonic formulation is based on solutions of the form

$$\mathbf{u}(x,t) = \widehat{\mathbf{u}}(x)\mathrm{e}^{-\mathrm{i}\omega t}, \tag{D-1}$$

where i is the imaginary unit $\mathrm{i}^2 = -1$, and ω is the angular frequency defined as $\omega = 2\pi f$, with f being the frequency in Hz. Equation (D-1) impacts the derivatives as follows

$$\begin{aligned}\frac{\partial \mathbf{u}(x,t)}{\partial t} &= -\mathrm{i}\omega\widehat{\mathbf{u}}(x)\mathrm{e}^{-\mathrm{i}\omega t},\\ \frac{\partial^2 \mathbf{u}(x,t)}{\partial t^2} &= -\omega^2\widehat{\mathbf{u}}(x)\mathrm{e}^{-\mathrm{i}\omega t}.\end{aligned}$$

The isotropic **elastic** wave-equation (C-1) is written in the time harmonic-formulation by substituting in Equation (D-1),

$$-\rho\omega^2\widehat{\mathbf{u}}\mathrm{e}^{-\mathrm{i}\omega t} - \nabla\left(\lambda\nabla\cdot\widehat{\mathbf{u}}\right)\mathrm{e}^{-\mathrm{i}\omega t} - \nabla\cdot\left(\mu\left[\nabla\widehat{\mathbf{u}} + (\nabla\widehat{\mathbf{u}})^T\right]\right)\mathrm{e}^{-\mathrm{i}\omega t} = 0$$

where the spatial dependency of λ, μ and $\mathbf{u}$ on x is suppressed in the notation here. This can be simplified by dividing through by $\mathrm{e}^{-\mathrm{i}\omega t}$, to give the time-harmonic isotropic elastic wave equation, which has no dependence on time,

$$-\rho\omega^2\widehat{\mathbf{u}} - \nabla\left(\lambda\nabla\cdot\widehat{\mathbf{u}}\right) - \nabla\cdot\left(\mu\left[\nabla\widehat{\mathbf{u}} + (\nabla\widehat{\mathbf{u}})^T\right]\right) = 0.$$

These steps can be reproduced for the **acoustic** wave equation (C-11). Again, we assume solutions have time-harmonic form, i.e.,

$$p(x,t) = \widehat{p}(x)\mathrm{e}^{-\mathrm{i}\omega t}. \tag{D-2}$$

Substituting this into (C-11) and dividing across by $\mathrm{e}^{-\mathrm{i}\omega t}$, we get

$$-\left(\frac{\omega^2}{c^2}+\nabla^2\right)\widehat{p}=0,$$

where we are writing the P-wave velocity $c_p(x)$ as $c(x)$ here, and the x dependence has been suppressed. This is the *Helmholtz* equation, and is usually written as

$$-\left(k+\nabla^2\right)\widehat{p}=0, \tag{D-3}$$

where the wavenumber k is the ratio of the angular frequency and the wavespeed

$$k(x,\omega)=\frac{\omega}{c(x)}. \tag{D-4}$$

We also define the index of refraction $n(x)$ by

$$n(x)=\frac{1}{c(x)}$$

and the squared-slowness, or model, which is the parameter used in acoustic full waveform inversion, by

$$m(x)=\frac{1}{c^2(x)}. \tag{D-5}$$

We note that the derivation of the Helmholtz equation with a source term is found in [72, Section 1.2.6]. In this thesis we mostly write the Helmholtz equation with a source term on the right-hand side and in terms of the model (D-5), in the following form,

$$-\left(\Delta u(x)+\omega^2\, m(x)\, u(x)\right)=q(x), \tag{D-6}$$

for some source term q. Note that we are writing $\widehat{p}$ as u here as this is the more standard notation.

Notes on Existence and Uniqueness

Now that the wave equations have been derived, we consider the concept of well-posedness. A problem is said to be well-posed if a solution exists (*existence*), the solution is unique (*uniqueness*), and the solution depends continuously on the data [48]. To ensure well-posedness, boundary conditions must be properly defined. One such boundary condition that ensures well-posedness is a radiation condition in the far-field. We discuss this radiation condition here and describe how it is implemented in practice.

Radiation Condition

In wave problems, radiation conditions are required at infinity to guarantee that the problem has a unique solution and, more generally, that the problem is well-posed. The radiation condition for the Helmholtz equation (D-3), is called the *Sommerfeld radiation*

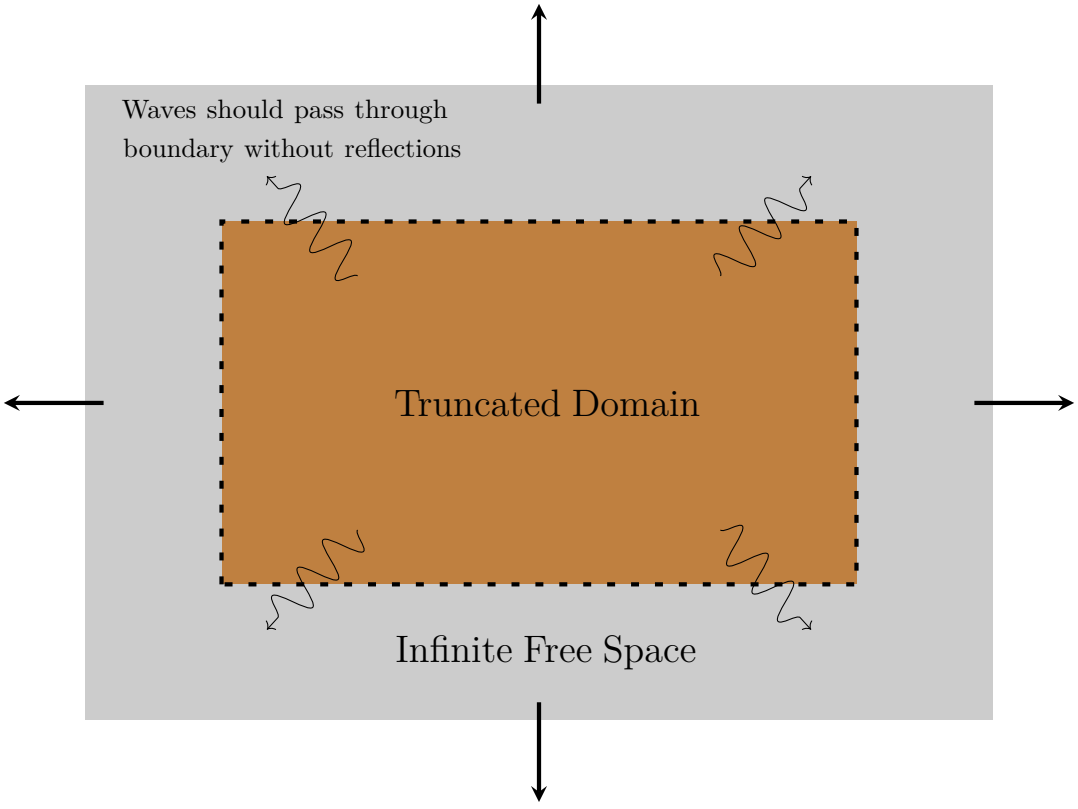
condition, and ensures that the system can lose its energy in the form of radiation but that no energy may be radiated from infinity into the system. This boundary condition is applied at infinity and rejects incoming ways, which do not make physical sense. In an infinite 3-dimensional domain, the condition has the form

$$\lim_{r\to\infty} r\left(\frac{\partial u}{\partial r} - \mathrm{i}ku\right) = 0, \tag{D-7}$$

where $r = |x|$ and k is the wave number. A detailed review of this boundary condition is avaiable in [158] and theorems about the well-posedness of the Helmholtz equation with radiation boundary conditions can be found in [48]. The radiation condition for the frequency domain elastic wave equation is discussed in [32].

Artificial Boundary and Absorbing Boundary Conditions

The Sommerfeld radiation condition applies to wave propagation problems on an infinite domain. However, to model wave phenomena numerically, the computational domain must be truncated to a finite domain.



Therefore, for implementations, the problem is reformulated through the introduction of an appropriate *artificial boundary*, with a new boundary condition to replace the Sommerfield radiation condition. Such boundary conditions attempt to reproduce the property that waves from inside the computational domain are passed through the

artificial boundary without generating spurious reflections back into the computational domain, hence ensuring a physically meaningful solution. However, to obtain this property perfectly is computationally expensive, so it is nearly always approximated.

Approximations of the Sommerfeld radiation condition which ensure the solution is as close as possible to the solution that would have been computed in the original physical domain are called Absorbing Boundary Conditions (ABCs). The aim of absorbing boundary conditions is to simulate the infinite domain of propagation by imposing a local boundary condition on the artificial boundary. ABCs were developed in 1977 by Engquist and Majda [66] in the time domain (and later in [20] in the frequency domain). In [66], an exact perfectly absorbing boundary condition is derived, which has the following form,

$$\frac{\partial u}{\partial n} - \mathcal{P}(u) = 0,$$

where $\mathcal{P}$ is a non-local operator. Unfortunately, this boundary condition is non-local and is therefore impractical from a computational point of view. Instead, approximations of the operator are made. The first-order approximation is equivalent to the impedance boundary condition introduced in Chapter 2,

$$\frac{\partial u}{\partial n} - \mathrm{i}\frac{\omega}{c}u = 0. \tag{D-8}$$

Note that this may also be written in terms of the model, since $\frac{1}{c} = \sqrt{m}$. Higher order approximations can also be made (see for example [66]).

We refer to [134] for a more detailed explanation and derivation of this technique and as well as other common approaches for solving wave propagation problems in a truncated domain, such as Perfectly Matched Layers (PMLs).

Appendix E

Discretisation of the Helmholtz Equation

In this appendix, we provide both a finite difference and finite element scheme for the discretisation of the Helmholtz equation, and then provide an example to show the connection between these two discretisation methods.

Finite Difference Scheme for the Helmholtz Equation

In this section, we provide a finite difference scheme for the following Helmholtz equation with impedance boundary conditions,

$$\begin{aligned} -(\Delta + \omega^2 m)u &= f \quad \text{on} \quad \Omega \\ \left(\frac{\partial}{\partial n} - \mathrm{i}\frac{\omega}{c}\right) u &= 0 \quad \text{on} \quad \partial\Omega \end{aligned}$$

where $u = u(x)$ and $m = m(x)$. The impedance boundary condition that we use is the same as that in (D-8), where we noted that $1/c = \sqrt{m}$.

There are two main steps in discretisation. First, the domain of interest is divided into a grid, and then, the equation is discretised. We go through these steps for the finite difference (FD) method, for a simple rectangular 2D domain.

Discretisation of the Domain

Consider a simple two-dimensional domain $\Omega \in \mathbb{R}^2$, with width L_x and height L_z. FD partitions the domain into a regular grid. We let N_x and N_z be the number of discretisation nodes in the x and z-direction respectively. We define the grid size in these directions as $h_x = L_x/(N_x - 1)$ and $h_z = L_z/(N_z - 1)$. The total number of discretisation points is $N_x \times N_z$. The nodes are labelled as (i, j) where i is the x index, from 0 to $N_x - 1$ and j stands for the z index, from 0 to $N_z - 1$. We write the discretised coordinates as $X_{i,j} = (x_i, z_j)$ where $x_i = ih_x$ and $z_j = jh_z$. Figure E-1 displays the discretisation grid.

Once the domain of interest has been discretised, the PDE is discretised and a discretised solution is sought at the nodal points.

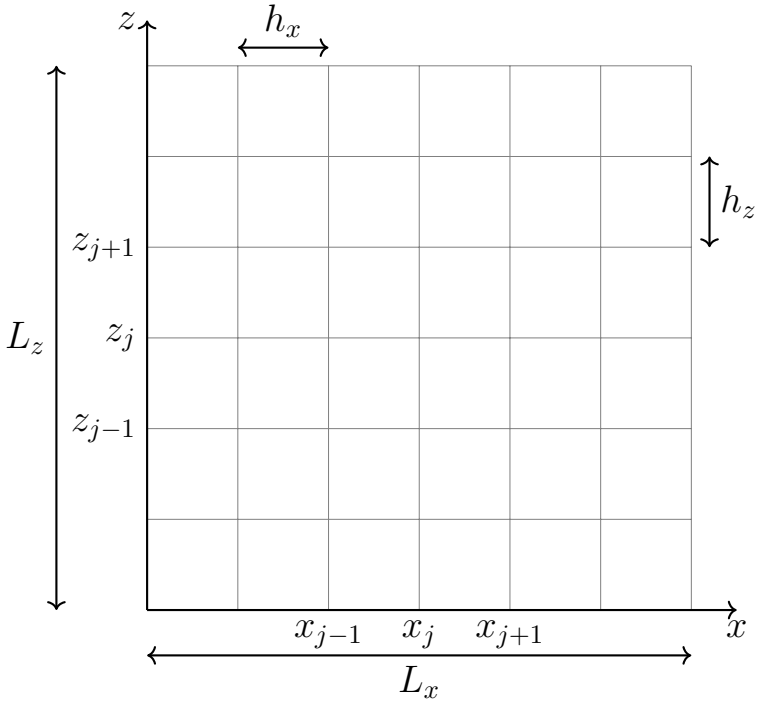


Figure E-1: *Discretised Grid.*

Discretisation of the PDE

In this section, we focus on discretising the Helmholtz equation in (E-1) with a first order finite difference method. The equation we are discretising is

$$-\Delta u(x) - \omega^2 \, m(x) \, u(x) = f(x), \tag{E-1}$$

where we have explicitly written the spatial dependence. We focus on (E-1) first and consider the boundary conditions later.

Finite differences aim to define the solutions at every node of the grid. In this case the discretised solution is defined as a column vector

$$\begin{aligned} \mathbf{u} &= [u(X_{0,0}), u(X_{0,1}), \ \dots \ u(X_{0,N_z-1}), u(X_{1,0}), \ \dots \ u(X_{i,j}), \ \dots \ u(X_{N_x-1,N_z-1})]^T, \\ &= [u_{0,0}, u_{0,1}, \ \dots \ u_{0,N_z-1}, u_{1,0}, \ \dots \ , u_{i,j}, \ \dots \ , u_{N_x-1,N_z-1}]^T, \end{aligned}$$

where $X_{i,j} = (x_i, z_i)$ indicates the two coordinates of the position at node (i, j) and the wavefield at that node is denoted $u_{i,j}$. Similarly, the model at node (i, j) is denoted $m_{i,j}$, and is arranged in a column vector

$$\boldsymbol{m} = [m_{0,0}, m_{0,1}, \ \dots \ m_{0,N_z-1}, m_{1,0}, \ \dots \ , m_{i,j}, \ \dots \ , m_{N_x-1,N_z-1}]^T. \tag{E-2}$$

Since we are working on a two-dimensional grid, we use the five-point formula to approximate the Laplacian, which is shown in Figure E-2. (We remark that more accurate

finite difference approximations can be obtained by using higher-order approximations. This leads to wider stencils.) Our discretised Laplacian term is then

$$\Delta u_{i,j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} + \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j-1}}{h_z^2}, \tag{E-3}$$

where $q_{i,j} = q(x_i, z_j), u_{i\pm1,j} = u(x_i \pm h_x, z_j)$, and $u_{i\pm1,j} = u(x_i, z_j \pm h_z)$. We note that in the case where $h = h_z = h_x$, this would simplify to

$$\Delta u_{i,j} = \frac{u_{i+1,j} + u_{i-1,j} - u_{i,j-1} + u_{i,j-1} - 4u_{i,j}}{h^2}. \tag{E-4}$$

The full approximation of (E-1) for *interior nodes* of the domain, using (E-3), is

$$-\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h_x^2} - \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j-1}}{h_z^2} - \omega^2 m_{i,j} u_{i,j} = f_{i,j}, \qquad X_{i,j} \in \Omega,\ X_{i,j} \notin \partial\Omega. \tag{E-5}$$

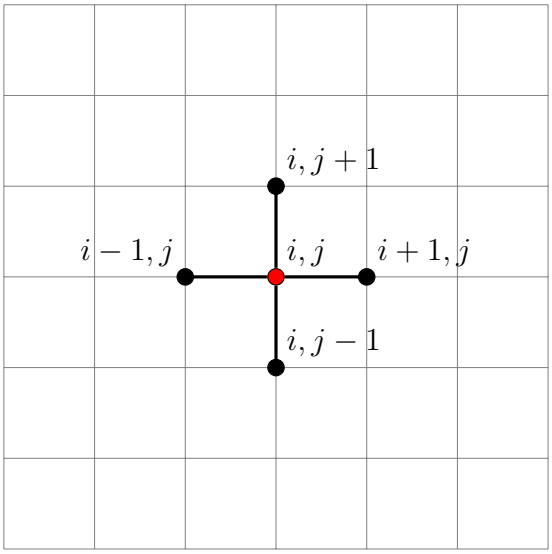


Figure E-2: *Stencil for 5-point discretisation of 2D Helmholtz operator.*

This discretization generates one equation per nodal point and each of them involves five grid points (except for when on the boundary) - the node itself and its four neighbours, as shown in Figure E-2.

Boundary conditions determine the approximation on the exterior nodes. There is a discussion of boundary conditions for the seismic problem in Appendix D. We demonstrate the finite difference procedure for the case where impedance boundary conditions (D-8) are applied on all boundaries of the domain. We restate (D-8) in terms of the model,

$$\frac{\partial u}{\partial n} - \mathrm{i}\omega\sqrt{m}u = 0 \qquad \text{on} \quad \partial\Omega, \tag{E-6}$$

where $\partial/\partial n$ is the normal derivative. The boundary $\partial\Omega$ corresponds to the node points $i = 0$ and $N_x - 1$, and $j = 0$ and $N_z - 1$. The boundary conditions at $i = 0$ (for any j) and $j = 0$ (for any i) are approximated with backward differences. For example, at z_0, we write the boundary condition as follows and use it to find the value of u at the fictitious point $(i, -1)$, which is outside the domain of the problem,

$$\left(-\frac{\partial u}{\partial z} - \mathrm{i}\omega\sqrt{m}u\right)_{i,0} = 0 \tag{E-7}$$

$$-\left(\frac{u_{i,0} - u_{i,-1}}{h_z}\right) - \mathrm{i}\omega\sqrt{m}_{i,0}u_{i,0} = 0$$

$$u_{i,-1} = (\mathrm{i}\omega\sqrt{m}_{i,0}\, h_z + 1)u_{i,0} \tag{E-8}$$

The fictitious point (E-8) can then be substituted into (E-5) at the boundary. The boundary condition at x_0 is similarly used to find the value of u at the fictitious point $(-1, j)$,

$$u_{-1,j} = (\mathrm{i}\omega\sqrt{m}_{0,j}\, h_x + 1)u_{0,j}.$$

At $i = N_x - 1$ (for any j) and $j = N_z - 1$ (for any i), the boundary conditions are approximated with forward differences and used to solve for the values of u at the fictitious points (N_x, j) and (i, N_z) in the same manner, giving

$$u_{N_x,j} = (\mathrm{i}\omega\sqrt{m}_{N_x-1,j}\, h_x + 1)u_{N_x-1,j},$$
$$u_{i,N_z} = (\mathrm{i}\omega\sqrt{m}_{i,N_z-1}\, h_z + 1)u_{i,N_z-1}.$$

Combining the discretised impedance boundary conditions with the discretised Helmholtz equation allows us to write the forward modelling equations as a linear matrix system,

$$A(\boldsymbol{m}, \omega)\mathbf{u} = \mathbf{f}, \qquad \mathbf{u}, \mathbf{f} \in \mathbb{C}^{N\times 1}, \quad A \in \mathbb{C}^{N\times N}$$

where $N = N_x N_z$, $\mathbf{u}$ contains all the wavefield values at nodal points $u_{i,j}$, $\mathbf{f}$ contains all the source values $f_{i,j}$ and A is the finite difference-matrix. The matrix A can be written as the sum of other matrices, such that

$$A(\boldsymbol{m}, \omega) = L - \omega^2 \operatorname{diag}(\boldsymbol{m}) - \mathrm{i}\omega\, B(\boldsymbol{m}). \tag{E-9}$$

where the matrix B is defined in (E-10) and the matrix L is defined in (E-11), and $\operatorname{diag}(\boldsymbol{m})$ denotes the diagonal matrix of size $N_x N_z \times N_x N_z$, with the model values (E-2) on the diagonal. The entries of the matrix B come from the imaginary parts of the impedance boundary condition and so are only non-zero on the boundaries. We define

the entry $B_{k,l}$, for $k = l = iN_z + j + 1$ as follows,

$$B_{k,l} = \begin{cases} \dfrac{\sqrt{m}_{i,j}}{h_x}, & \text{when} \quad i = \{0,\, N_x - 1\} \quad \forall j \notin \{0, N_z - 1\} \\ \dfrac{\sqrt{m}_{i,j}}{h_z}, & \text{when} \quad j = \{0,\, N_z - 1\} \quad \forall i \notin \{0, N_x - 1\}, \\ \dfrac{\sqrt{m}_{i,j}}{h_x} + \dfrac{\sqrt{m}_{i,j}}{h_z}, & \text{when} \quad i = \{0,\, N_x - 1\} \quad \text{and} \quad j = \{0, N_z - 1\}, \\ 0, & \text{else} \end{cases} . \tag{E-10}$$

The matrix L can be written compactly using the kronecker product,

$$L = -\left(\mathbf{D}_{xx} \otimes \mathbf{I}_{N_z} + \mathbf{I}_{N_x} \otimes \mathbf{D}_{zz}\right), \tag{E-11}$$

where $\mathbf{I}_{N_x}$ and $\mathbf{I}_{N_z}$ are the identity matrices of size $N_x \times N_x$ and $N_z \times N_z$ respectively and $\mathbf{D}_{xx}$ and $\mathbf{D}_{zz}$ are close to the 1D finite difference Laplacians in the x and z direction respectively, but incorporate the additional real parts of the boundary condition terms. These matrices are defined below in (E-12) and (E-13). The derivative matrix in the x direction is of size $N_x \times N_x$ and the elements are

$$[\mathbf{D}_{xx}]_{k,l} = \frac{1}{h_x^2} \begin{cases} -2 & \text{when} \quad k = l \notin \{1, N_x\}, \\ -1 & \text{when} \quad k = l \in \{1, N_x\}, \\ 1 & \text{when} \quad k = l + 1, \quad l \neq N_x, \\ 1 & \text{when} \quad k = l - 1, \quad l \neq 1. \end{cases} \tag{E-12}$$

The derivative in the z direction is of size $N_z \times N_z$ and similarly has a tridiagonal form, with entries

$$[\mathbf{D}_{zz}]_{k,l} = \frac{1}{h_z^2} \begin{cases} -2 & \text{when} \quad k = l \notin \{1, N_z\}, \\ -1 & \text{when} \quad k = l \in \{1, N_z\}, \\ 1 & \text{when} \quad k = l + 1, \quad l \neq N_z, \\ 1 & \text{when} \quad k = l - 1, \quad l \neq 1. \end{cases} \tag{E-13}$$

Remark E-1. *The finite difference discretisation presented here is low order but is sufficient for implementations in this thesis since all experiments involve relatively low frequencies that allow this discretisation to remain accurate. For higher frequency problems, higher-order discretisation schemes are preferred.*

Finite Element Discretisation of Helmholtz Problem

Here we present the Helmholtz equation with impedance boundary conditions (2.2.5) in its discrete form using finite elements, and then we present an example that makes the link between this discretisation and the finite difference discretisation.

We start by writing the problem (2.2.5) in weak form, i.e., find $u \in H^1(\Omega)$ such that

$$a(u, v) = F(v), \quad \text{for all } v \in H^1(\Omega)$$

where $a(u, v)$ is defined as in (2.2.9),

$$a(u, v) = \int_\Omega \left(\nabla u \cdot \nabla \overline{v} - \omega^2 m \, u \overline{v} \right) - \mathrm{i}\omega \int_{\partial\Omega} u \overline{v}$$

and $F(v)$ is defined as

$$F(v) = \int_\Omega f \overline{v}.$$

Given a finite dimensional subspace $V_N \subset H^1(\Omega)$, the finite element method seeks $u^N \in V_N$ such that $a(u^N, v^N) = F(v^N)$ for all $v^N \in V_N$. Let $\{\Phi_i : i \in \{1, \ldots, N\}\}$ be a basis for V_N, and let $u^N = \sum_{i=1}^N u_i \Phi_i$. The linear system that arises from the FEM discretisation is

$$A(\boldsymbol{m}, \omega)\mathbf{u} = \mathbf{f} \tag{E-14}$$

where $A \in \mathbb{C}^{N \times N}$, $\mathbf{u} \in \mathbb{C}^{N \times 1}$, and $\mathbf{f} \in \mathbb{C}^{N \times 1}$. The matrix A can be written as a sum of three other matrices

$$A(\boldsymbol{m}, \omega) = S - \omega^2 K(\boldsymbol{m}) - \mathrm{i}\omega C \tag{E-15}$$

where the matrices and vectors appearing in (E-14) and (E-15) are defined as follows:

$$S_{i,j} = \int_\Omega \nabla \Phi_i . \overline{\nabla \Phi_j}, \qquad K(\boldsymbol{m})_{i,j} = \int_\Omega m \Phi_i \overline{\Phi}_j,$$
$$C_{i,j} = \int_{\partial\Omega} \Phi_i \overline{\Phi}_j, \qquad \mathbf{f}_i = \int_\Omega f \overline{\Phi}_j, \qquad (\mathbf{u})_i = u_i,$$

where $\boldsymbol{m}$ and m are defined in Table 2.2.1. The matrix S is called the stiffness matrix, $K(\boldsymbol{m})$ is the domain mass matrix and C is the boundary mass matrix. The matrix A is symmetric but not Hermitian due to the i in front of the C matrix, which appears due to the impedance boundary condition. There is a separate large sparse system of linear equations of the form (E-14) for each frequency ω and for each source. We have stated (E-14) in a general form without too many specific details to emphasise that the method of discretisation presented is very flexible.

Example Demonstrating the Link between Finite Element and Finite Difference Discretisation: We now state some more details of the finite element discretisation method in the case when the domain Ω is the unit square $[0, 1] \times [0, 1]$ and the basis functions are piecewise linear on a uniform mesh of grid size $h = 1/(n-1)$. Thus, $N = n^2$, and the mesh has nodes $X_{i,j} = (x_i, z_j) = (ih, jh)$, $i, j = 0, \ldots, n-1$, with $h = 1/(n-1)$. The mesh consists of triangles and a patch of this mesh is depicted in Figure E-3. We choose linear hat functions as the basis functions, i.e., $\Phi_{ij} = 1$ at

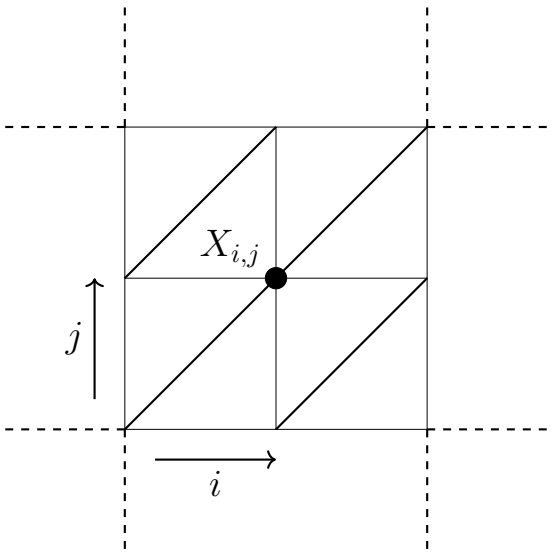


Figure E-3: *Section of Finite Element Mesh*

node X_{ij} and $\Phi_{ij} = 0$ at all other nodes. It is well-known that in this case, when X_{ij} is an interior node, the stiffness matrix S takes the form

$$S_{(i,j),(i',j')} = \begin{cases} 4 & \text{when } (i',j') = (i,j) \\ -1 & \text{when } (i',j') = (i-1,j),(i-1,j),(i,j-1),(i,j+1). \end{cases}$$

Therefore, $h^{-2}S$ coincides with the '5-point stencil' of the finite difference approximation (which is discussed in the finite difference section above). For the domain mass matrix we have

$$K(\boldsymbol{m})_{(i,j)(i',j')} = \int_\Omega m\, \Phi_{ij}\Phi_{i'j'}.$$

In general, this domain mass matrix is different to the corresponding finite difference approximation, however, we can simplify it by applying the quadrature rule on each element τ,

$$\int_\Omega f \approx \frac{\text{area}(\tau)}{3} \sum_{v\in\tau} f(v)$$

where the sum is over the three vertices $v \in \tau$. Since $\text{area}(\tau) = \frac{h^2}{2}$ and, when $X_{i,j}$ is an interior node, there are six triangles touching the interior node, we obtain,

$$\begin{aligned} K(\boldsymbol{m})_{(i,j)(i,j)} &\approx 6\frac{h^2}{6} m(X_{ij})\Phi(X_{ij})^2, \\ &= h^2 m(X_{ij}), \end{aligned}$$

and

$$K(\boldsymbol{m})_{(i,j)(i',j')} \approx 0 \qquad \text{when } (i,j) \neq (i',j').$$

At interior nodes, after scaling by h^{-2}, the finite element method with triangular elements coincides with the finite difference method detailed above. (We do not write the details of the scheme at the boundary nodes here and only give the precise formula for the finite difference case.)

Appendix F

Optimisation

In this thesis we use local optimisation methods for finding the parameters that minimise our objective functions. In particular, we use *gradient-based* local optimisation methods. Gradient-based optimisation algorithms are widely used for solving a variety of optimisation problems, because these techniques can be efficient and they can solve problems with large numbers of variables. However, local optimisation methods have some drawbacks, which include that they can only locate a local optimum, and that they have difficulty dealing with discontinuous functions.

In this section, we review local optimisation methods that are popular in FWI. Therefore, we describe these methods in the context finding an optimal model but the theory is kept general so that it applies to any optimisation problem. The theory presented here is largely based on information in [136].

Local optimisation methods involve making a starting guess, $\boldsymbol{m}_0$, and iteratively updating the model $\boldsymbol{m}$ by searching the local model space. The iterative update for local optimisation algorithms can be generally be written mathematically as follows,

$$\boldsymbol{m}_{k+1} = \boldsymbol{m}_k + \alpha_k \mathbf{d}_k, \tag{F-1}$$

where k is the iteration number, $\mathbf{d}$ is referred to as a descent direction, and α is a step size. At iteration k, one finds $\mathbf{d}_k$ using gradient (or sometimes Hessian) information. Then, a step size α_k is found that controls how far one moves in the descent direction. A value for α may be found using a line search method, which we discuss later. The following sections discuss specific methods for computing both the descent direction and step size.

Newton's Method

The first optimisation method that we discuss is Newton's method. This method is important as we see later that many other optimisation methods are based on an approximation of Newton's method.

Newton's method is derived from the first order Taylor expansion. Considering a model $\boldsymbol{m} \in \mathbb{R}^M$, and an objective function $\phi(\boldsymbol{m})$ that is regular enough, and assuming a small perturbation $\delta\boldsymbol{m}$, we can expand the function as follows

$$\phi(\boldsymbol{m} + \delta\boldsymbol{m}) = \phi(\boldsymbol{m}) + \delta\boldsymbol{m}^T \nabla_{\boldsymbol{m}} \phi(\boldsymbol{m}) + \frac{1}{2} \delta\boldsymbol{m}^T H(\boldsymbol{m}) \delta\boldsymbol{m} + \mathcal{O}(|\delta\boldsymbol{m}|^3), \tag{F-2}$$

where we have truncated the expansion at second order, leaving a residual of $\mathcal{O}(|\delta \boldsymbol{m}|^3)$ The Hessian matrix $H(\boldsymbol{m})$ is the $M \times M$ second derivative matrix, whose elements are given by

$$H_{ij}(\boldsymbol{m}) = \frac{\partial^2 \phi(\boldsymbol{m})}{\partial m_i \partial m_j} \qquad \text{for} \quad i, j = (1, 2, .., M).$$

We want to find the *descent direction* $\delta \boldsymbol{m}$ that will locate the minimum of the quadratic approximation in (F-2). A descent direction is a direction $\delta \boldsymbol{m}$ such that $\langle \delta \boldsymbol{m}, \nabla_{\boldsymbol{m}} \phi(\boldsymbol{m}) \rangle < 0$, and thus $\phi(\boldsymbol{m} + \delta \boldsymbol{m}) < \phi(\boldsymbol{m})$. When the minimum of the objective function is reached, then its gradient is zero. Therefore, to solve for the minimiser, we take the derivative of (F-2) with respect to $\delta \boldsymbol{m}$ and set it to zero. Neglecting higher order terms, we find that the solution is given by

$$\delta \boldsymbol{m} = -H^{-1}(\boldsymbol{m}) \nabla_{\boldsymbol{m}} \phi(\boldsymbol{m}); \tag{F-3}$$

this is termed the *Newton step*. It is important to note that, since the Hessian matrix may not always be positive definite, the Newton step may not always be a descent direction.

The Newton method aims in building a sequence of $\boldsymbol{m}_k$ converging towards the zero of the function. At iteration k, the iterative update is given as

$$\boldsymbol{m}_{k+1} = \boldsymbol{m}_k - H_k^{-1} \nabla_{\boldsymbol{m}} \phi_k, \tag{F-4}$$

where $H_k = H(\boldsymbol{m}_k)$ and $\nabla_{\boldsymbol{m}} \phi_k = \nabla_{\boldsymbol{m}} \phi(\boldsymbol{m}_k)$. The iterations are performed until some convergence criteria is reached. Note that in its classic form, Newton's method uses a constant step size of $\alpha = 1$. However, sometimes in practice the method is modified to include a line search to ensure that the update in (F-4) is a descent direction and to improve the efficiency of the method. For a discussion of Newton's method specifically in FWI see [150].

Newton's method has a quadratic rate of convergence (under certain conditions), which is highly desirable. However, the computational cost involved in computing the Hessian makes this method impractical in many applications. Therefore, in practice, methods that are based on Newton's method are more popular than Newton's method itself, such as methods which use the first-order part of the Hessian only (Gauss-Newton Method) or an approximation of the Hessian (Quasi-Newton Methods).

Remark F-1. *During the derivation of Newton's method, we used the fact that the gradient of the objective function is zero at the global minimum. However, we note that the gradient is also zero for any local minimum, or maximum, of the function, and so the condition $\nabla \phi = 0$ is necessary but not sufficient for characterising the global minimum, unless the objective function is strictly or strongly convex.*

Steepest Descent

The steepest descent approach follows the negative of the gradient of the objective function to find its minimum. The descent direction for the steepest-descent method is

$$\mathbf{d}_k^{SD} = -\nabla \phi(\boldsymbol{m}_k), \tag{F-5}$$

which is substituted into (F-1) to give the steepest descent method. (Note that we write $\nabla = \nabla_{\boldsymbol{m}}$ in the rest of this appendix to simplify the notation.) The step size in (F-1) may be chosen using a line search. The steepest descent method's advantage is in its simplicity, since it only requires the computation of the first derivative of the current step. However, it has the disadvantage that it can be extremely slow on difficult problems. More sophisticated algorithms are generally used in FWI.

Quasi-Newton Methods

A typical iteration of a Quasi-Newton method has the form of (F-1), with descent direction,

$$\mathbf{d}_k^{QN} = -B_k^{-1}\nabla\phi(\boldsymbol{m}_k), \tag{F-6}$$

where B_k is a positive definite matrix, updated from iteration to iteration, chosen so that (F-6) is an approximation to the Newton step (F-3), i.e., B_k is an approximation to the Hessian $H(\boldsymbol{m}_k)$. Quasi-Newton methods, like steepest descent, require only the first-order information to be supplied at each iterate. However, the improvement in convergence over steepest descent can be dramatic. Superlinear convergence of a Quasi-Newton method is guaranteed under the Dennis-Moré condition ([62], [136, Section 8.4]).

What makes a quasi-Newton method work so well is the choice of the matrix B_k at each iteration. At each step of a Quasi-Newton method, the aim is to find $B_k \approx H(\boldsymbol{m}_k)$. The main idea behind this approximation is to use model and gradient information from current and past iterates. Two successive iterates $\boldsymbol{m}_k$ and $\boldsymbol{m}_{k+1}$, and successive gradients $\nabla\phi(\boldsymbol{m}_k)$ and $\nabla\phi(\boldsymbol{m}_{k+1})$ provide information about the Hessian matrix, since

$$\nabla\phi(\boldsymbol{m}_{k+1}) - \nabla\phi(\boldsymbol{m}_k) \approx H(\boldsymbol{m}_k)(\boldsymbol{m}_{k+1} - \boldsymbol{m}_k),$$

where the approximation would be an equality if the function we are dealing with is quadratic. In addition, the approximation tends towards an equality as $||\boldsymbol{m}_{k+1} - \boldsymbol{m}_k|| \to 0$. Therefore, at every iteration, B_{k+1} is chosen to satisfy

$$B_{k+1}\mathbf{s}_k = \mathbf{y}_k, \tag{F-7}$$

where

$$\mathbf{s}_k = \boldsymbol{m}_{k+1} - \boldsymbol{m}_k, \qquad \mathbf{y}_k = \nabla\phi(\boldsymbol{m}_{k+1}) - \nabla\phi(\boldsymbol{m}_k).$$

Equation (F-7) is called the secant equation, or Quasi-Newton condition.

One of the most popular choices for B_k is the BFGS formula, which is

$$B_{k+1} = B_k - \frac{B_k\mathbf{s}_k\mathbf{s}_k^T B_k}{\mathbf{s}_k^T B_k \mathbf{s}_k} + \frac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{s}_k}, \tag{F-8}$$

where B_0 is an initial approximation to the Hessian that should be symmetric and positive definite. The matrix B_0 is sometimes chosen as the identity matrix. In this

case, the first descent direction (F-6) of the BFGS method is equivalent to the steepest descent direction. It is important to note that to ensure the BFGS matrix B_k is positive definite, the step size in (F-1) should be chosen via Wolfe Line Search. This is also noted later in Remark F-2, after we give details of the Wolfe conditions.

Instead of computing B_{k+1} and then solving (F-6) for the descent direction, practical BFGS algorithms will instead update and store and approximation to the *inverse* Hessian, B_{k+1}^{-1}. The matrix-vector product is cheaper to compute than solving a linear system. The inverse of B_{k+1} can be obtained efficiently by applying the Sherman–Morrison formula

$$B_{k+1}^{-1} = \left(I - \rho_k \mathbf{s}_k \mathbf{y}_k^T\right) B_k^{-1} \left(I - \rho_k \mathbf{y}_k \mathbf{s}_k^T\right) + \rho_k \mathbf{s}_k \mathbf{s}_k^T, \qquad \rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}, \tag{F-9}$$

where I is the identity matrix. For ease of notation, in the rest of this section, we denote B_k^{-1} as $\widetilde{H}_k$, so that (F-7) may also be written as $\mathbf{s}_k = \widetilde{H}_{k+1}\mathbf{y}_k$.

The BFGS method involves storing the $M \times M$ approximation to the inverse Hessian (recall that M is the length of $\boldsymbol{m}$). This can become infeasible to store this for large M. This issue can be addressed by using a limited memory version of the BFGS method, which we discuss next.

Limited-memory BFGS

The Limited-memory BFGS, or L-BFGS, algorithm modifies BFGS to obtain inverse Hessian approximations that can be stored in just a few vectors of length M. The key idea of L-BFGS is that instead of storing a large $M \times M$ approximation, it stores just n vectors of length M that implicitly represent the approximation, where n is chosen by the user such that $n \ll M$. It uses information from the n most recent iterations only, since the information from earlier iterations is considered to be less likely to be relevant to the Hessian behaviour at the current iteration, and so is discarded to save memory. In other words, the information stored is the last n pairs of $\{\mathbf{s}_k, \mathbf{y}_k\}$, which means that the algorithm needs $2 \times n \times M$ storage (instead of $M \times M$ for BFGS).

The L-BFGS implementation is now discussed in more detail. At iteration k, the descent direction is computed by performing a sequence of operations with the stored set of vector pairs $\{\mathbf{s}_i, \mathbf{y}_i\}$ for $i = k-n, \ldots, k-1$. The first step of the algorithm is to choose a temporary initial guess $\widetilde{H}_k^0$ (which differs to the standard BFGS iteration, as this initial approximation will be allowed to vary from iteration to iteration). By repeated application of (F-9),the L-BFGS inverse Hessian approximation $\widetilde{H}_k$ satisfies,

$$\begin{aligned}\widetilde{H}_k = \left(V_{k-1}^T ... V_{k-n}^T\right) \tilde{H}_k^0 \left(V_{k-n} ... V_{k-1}\right) + \rho_{k-n} \left(V_{k-1} ... V_{k-n+1}\right) \mathbf{s}_{k-n} \mathbf{s}_{k-n}^T \left(V_{k-n+1}^T ... V_{k-1}^T\right) + \\ \rho_{k-n+1} \left(V_{k-1} ... V_{k-n+2}\right) \mathbf{s}_{k-n+1} \mathbf{s}_{k-n+1}^T \left(V_{k-n+2}^T ... V_{k-1}^T\right) + \ ... \ + \rho_{k-1} \mathbf{s}_{k-1} \mathbf{s}_{k-1}^T\end{aligned} \tag{F-10}$$

where we have written $V_i = I - \rho_k \mathbf{y}_i \mathbf{s}_i^T$ (see also [136, Equation (9.5)]). Therefore the computation of $\widetilde{H}_k \nabla \phi_k$ may be done recursively. The procedure for this recursive computation is shown in Algorithm F-2.

The full L-BFGS algorithm is shown in Algorithm F-1. We see from this algorithm that, once we update the model, the oldest element in $\{\mathbf{s}_i, \mathbf{y}_i\}$ is replaced by $(\mathbf{s}_k, \mathbf{y}_k)$. The algorithm requires an initial guess for $\widetilde{H}_k^0$, which for computational simplicity is often chosen to be a multiple of the diagonal. The choice of matrix $\widetilde{H}_k^0$ is allowed to vary between iterations. An effective initial choice is $\tilde{H}_k^0 = \gamma_k I$, where

$$\gamma_k = \frac{\mathbf{s}_{k-1}^T \mathbf{y}_{k-1}}{\mathbf{y}_{k-1}^T \mathbf{y}_{k-1}}. \tag{F-11}$$

(see [136, pages 200, 201]).

The main advantage of the L-BFGS method is that it is relatively inexpensive. The work per iteration is $\mathcal{O}(nM)$, while for the BFGS method this is $\mathcal{O}(M^2)$. A drawback of L-BFGS is that the optimal choice of n is problem dependent.

Algorithm F-1 L-BFGS

Inputs: Starting guess $\boldsymbol{m}_0$, Memory integer n
Set $k = 0$
Repeat
    Choose $\widetilde{H}_k^0$ (e.g., by (F-11))
    Compute $\mathbf{d}_k$ from `Algorithm` F-2
    $\alpha_k \leftarrow$ Line Search
    $\boldsymbol{m}_{k+1} \leftarrow \boldsymbol{m}_k + \alpha_k \mathbf{d}_k$
    if $k > n$ **then**
        Discard pair $\left\{\mathbf{s}_{k-n}, \mathbf{y}_{k-n}\right\}$ from storage
    end if
    Compute and store $\mathbf{s}_k = \boldsymbol{m}_{k+1} - \boldsymbol{m}_k$, $\mathbf{y}_k = \nabla\phi(\boldsymbol{m}_{k+1}) - \nabla\phi(\boldsymbol{m}_k)$
    $k = k + 1$
until convergence
Output: Minimum $\widehat{\boldsymbol{m}}$

Algorithm F-2 L-BFGS update using two-loop recursion

Inputs: Initial approximation $\widetilde{H}_k^0$, gradient at current iterate $\nabla\phi(\boldsymbol{m}_k)$, Stored vectors $\mathbf{s}, \mathbf{y}$
$\mathbf{r} \leftarrow \nabla\phi(\boldsymbol{m}_k)$
for $i = k-1$ **to** $k-n$ **do**
    $\alpha_i \leftarrow \rho_i \mathbf{s}_i^T \mathbf{r}$
    $\mathbf{r} \leftarrow \mathbf{r} - \alpha_i \mathbf{y}_i$
end for
$\mathbf{r} \leftarrow \widetilde{H}_k^0 \mathbf{r}$
for $i = k-n$ **to** $k-1$ **do**
    $\beta \leftarrow \rho_i \mathbf{y}_i^T \mathbf{r}$
    $\mathbf{r} \leftarrow \mathbf{r} + \mathbf{s}_i(\alpha_i - \beta)$
end for
Output: $\mathbf{d}_k = -\mathbf{r}$

The Gauss-Newton Method

Gauss-Newton methods are a modification of Newton's method (F-4). Instead of computing the full Hessian to find the descent direction, like in (F-3), the second-order derivative terms are excluded. To demonstrate what this means, consider the function we want to minimise in the form

$$\phi(\boldsymbol{m}) = \frac{1}{2}||\boldsymbol{\varepsilon}(\boldsymbol{m})||_2^2. \tag{F-12}$$

The residual is arranged in a vector $\boldsymbol{\varepsilon} = [\varepsilon_1\ \varepsilon_2\ , \dots\ , \varepsilon_{N_r}]^T$ and so the objective function may also be written as $\phi(\boldsymbol{m}) = \frac{1}{2}\sum_{i=1}^{N_r} \varepsilon_i^2(\boldsymbol{m})$.

The derivatives of ϕ can be expressed in terms of the *Jacobian* of the residual $\boldsymbol{\varepsilon}$, which is the $N_r \times M$ matrix of partial derivatives defined by

$$J(\boldsymbol{m}) = \left[\frac{\partial \varepsilon_i}{\partial m_j}\right]_{j=1,..,M\,,\,i=1,..,N_r} \tag{F-13}$$

Namely, the derivatives of ϕ with respect to $\boldsymbol{m}$ may be expressed as

$$\nabla\phi(\boldsymbol{m}) = \sum_{i=1}^{N_r} \varepsilon_i(\boldsymbol{m})\nabla\varepsilon_i(\boldsymbol{m}) = J(\boldsymbol{m})^T\boldsymbol{\varepsilon}(\boldsymbol{m}), \tag{F-14}$$

$$\begin{aligned} H(\boldsymbol{m}) = \nabla^2\phi(\boldsymbol{m}) &= \sum_{i=1}^{N_r} \nabla\varepsilon_i(\boldsymbol{m})^T\nabla\varepsilon_i(\boldsymbol{m}) + \sum_{i=1}^{N_r} \varepsilon_i(\boldsymbol{m})\nabla^2\varepsilon_i(\boldsymbol{m}) \\ &= J(\boldsymbol{m})^T J(\boldsymbol{m}) + \sum_{i=1}^{N_r} \varepsilon_i(\boldsymbol{m})\nabla^2\varepsilon_i(\boldsymbol{m}) \end{aligned} \tag{F-15}$$

The Gauss-Newton method involves approximating the Hessian by the first order term in (F-15),

$$H^{GN}(\boldsymbol{m}) = J(\boldsymbol{m})^T J(\boldsymbol{m}), \tag{F-16}$$

so that the descent direction for this method is

$$\begin{aligned}\mathbf{d}_k^{GN} &= -\left(J(\boldsymbol{m}_k)^T J(\boldsymbol{m}_k)\right)^{-1} \nabla\phi(\boldsymbol{m}_k), \\ &= -\left(J(\boldsymbol{m}_k)^T J(\boldsymbol{m}_k)\right)^{-1} J(\boldsymbol{m}_k)^T \boldsymbol{\varepsilon}(\boldsymbol{m}_k).\end{aligned} \tag{F-17}$$

which is then used in (F-1) with line search. Note that (F-17) is a descent direction under the assumption that J is full rank, meaning H^{GN} is a positive definite matrix.

This modification gives some advantages over Newton's method. The approximation $H \approx H^{GN}$ saves the time and effort involved in computing the second order term $(\nabla^2\varepsilon_i(\boldsymbol{m}),\ i = 1, \ldots, N)$. In addition, if the Jacobian is explicitly computed in the computation of the gradient (F-14), then the approximation (F-16) is almost free. There are also some cases where the approximation (F-16) is accurate, and so the Gauss-Newton method has a similar performance to Newton's method. This happens when the first term in (F-15) dominates over the second term, for example, when the values of the residual ε_i are small (small residual case), or when each ε_i is nearly linear, so that its second derivatives are small. In practice, many least-squares problems of the form (F-12) have small, or zero, residuals at the minimum, and so the Gauss-Newton approximation becomes accurate as the minimum is approached, and rapid local convergence is observed on these problems. The speed of convergence in general depends on how much the term $J^T J$ dominates the second term in the Hessian.

However, if J is not full rank, the Gauss-Newton method will fail, as $J^T J$ becomes singular. This problem is discussed in the context of FWI in Appendix G.

The Levenberg-Marquadt Method

We now discuss a further modification of Newton's methods/Gauss-Newton's method.

If the Hessian is close to singular, the inverted Hessian in (F-4) can be numerically unstable, and Newton's method may diverge from the solution. An example of how this may be overcome is through modifications of the Hessian (see [136, Section 6.3]). The Hessian can be modified by adding a correction matrix E_k to make $H(\boldsymbol{m}_k) + E_k$ positive definite. The simplest way of choosing E_k would be to find a scalar $\tau > 0$ such that $H(\boldsymbol{m}_k) + \tau I$ is positive definite. The descent direction in this case is written as

$$\mathbf{d}_k^{damp} = -\left(H(\boldsymbol{m}_k) + \tau_k I\right)^{-1} \nabla\phi(\boldsymbol{m}_k), \tag{F-18}$$

where τ_k is the τ parameter at iteration k. The parameter τ may be held constant or varied as the iteration proceeds.

Levenberg [114] and Marquadt [122] proposed a similar idea for the Gauss-Newton method, to address the difficulties associated with a singular Jacobian mentioned earlier.

Their idea is now known as the Levenberg-Marquadt method. Levenberg's original idea was to replace the Gauss-Newton descent direction with a 'damped' version, as follows,

$$\begin{aligned}\mathbf{d}_k^{LM} &= -\left(H^{GN}(\boldsymbol{m}_k) + \tau_k I\right)^{-1} \nabla\phi(\boldsymbol{m}_k) \\ &= -\left(J(\boldsymbol{m}_k)^T J(\boldsymbol{m}_k) + \tau_k I\right)^{-1} \nabla\phi(\boldsymbol{m}_k). \end{aligned} \tag{F-19}$$

The parameter $\tau > 0$, often referred to as the damping factor, may be updated from iteration to iteration. This parameter ensures the descent direction is well-defined, even when the Jacobian is singular.

The Levenberg-Marquadt method can be considered as an 'interpolation' between the Gauss-Newton method and the gradient descent method. For large values of τ, (F-19) approaches

$$\mathbf{d}_k = -\frac{1}{\tau}\nabla\phi(\boldsymbol{m}_k),$$

which is like a small step in the steepest descent direction. The Levenberg-Marquadt method will therefore behave like steepest descent with step size $1/\tau$. This results in slower convergence but is useful when the current iterate is far from the solution. If τ is very small, then $\mathbf{d}^{LM} \approx \mathbf{d}^{GN}$, which is a good step in the final stages of the iterations (as long as H^{GN} is not close to singular), since when the residual $\boldsymbol{\varepsilon}$ in (F-12) gets small, the Gauss-Newton Hessian is a good approximation of the Hessian. In the case where J is rank-deficient, τ should not go to zero. In [68], it is suggested to chose $\tau_k = ||\boldsymbol{\varepsilon}(\boldsymbol{m}_k)||_2^\delta$, for some $\delta \in [1, 2]$, when minimising a function of the form (F-12). It is proven that under some assumptions (e.g., $\boldsymbol{\varepsilon}$ being continuously differentiable, J being Lipschitz continuous), the Levenberg-Marquadt method with this choice of τ and without line search will converge quadratically. Note that the damping parameter influences both the direction and the size of the step, and so this method is used without a specific line search (i.e., we don't need to compute α_k in (F-1)).

A disadvantage of the Levenberg-Marquadt method is that, when τ is large, and the method tends to steepest descent, Hessian information is not used at all. To avoid slow convergence, a modified Levenberg-Marquadt can be used instead, where

$$\mathbf{d}_k^{LM} = -\left(J(\boldsymbol{m}_k)^T J(\boldsymbol{m}_k) + \tau_k \,\mathrm{diag}(J(\boldsymbol{m}_k)^T J(\boldsymbol{m}_k))\right)^{-1} \nabla\phi(\boldsymbol{m}_k), \tag{F-20}$$

where the identity matrix has been replaced by the diagonal elements of the approximate $J^T J$. Therefore, even when τ is large, we still get some benefit from the Hessian approximation.

In terms of FWI, forms of the Levenberg-Marquadt method are used often, to avoid to computation of the full Hessian, and also because the Gauss-Newton Hessian in FWI is generally ill-conditioned. For example, this method is used in [139], where the damping parameter τ_k in (F-19) is defined as

$$\tau_k = \frac{n}{100} \times \max(H), \tag{F-21}$$

where $\max(H)$ indicates the maximum value of the approximate Hessian being used, and n is a percentage to be chosen. In the initial steps of the algorithm $n = 10$ is

suggested, and this is reduced as iterations proceed. The Levenberg-Marquadt method also appears in, for example in [150, Equation 36] and [128, Section 4.2.2] where a connection is made with the trust region method.

Stopping Criteria

The outlined minimisation algorithms require some stopping criteria to indicate convergence. We generally want to stop at iteration k if one of the following situations occurs:

- $||\nabla\phi_k|| < \text{tol}_1$
- $||\phi_k - \widetilde{\phi}|| < \text{tol}_2$
- $\dfrac{||\boldsymbol{m}_k - \boldsymbol{m}_{k-1}||}{||\boldsymbol{m}_k||} < \text{tol}_3$
- $||\phi_{k-1} - \phi_k|| < \text{tol}_4$

where $||.||$ is some chosen norm. The reasoning behind the first stopping criterion is that the gradient of the objective function at a minimum is zero. Therefore, when the absolute value of the gradient becomes sufficiently small during the algorithm, the algorithm can be terminated. The second stopping criteria can be used when there is a known minimum value of the objective function $\tilde{\phi}$. The algorithm will terminate when the value of objective function becomes close enough to the known minimum value. The third stopping criterion is satisfied when the updates to the parameters have stagnated, and indicates that further iterations would be a waste of computational time. Similarly, the final condition saves on computational expense by preventing continuing iterations when the reduction in the objective function is excessively small. The third and fourth conditions are often implemented with some minimum iteration number check, to avoid premature stopping, i.e., the algorithm is stopped only if these inequalities hold for a specific number of previous iterations. In addition, the above stopping criteria are generally implemented with a further condition that stops the iterations once a maximum iteration count has been reached.

The tolerance is chosen depending on the problem. If the tolerance is too small, the problem may end up being 'over-solved' by fitting the noise in the data or by making trivially small steps at a large computational cost. If the tolerance is too large, convergence happens too early and the computed solution is far from the true solution.

Line Search

So far, we have discussed the different methods for computing the descent direction $\mathbf{d}_k$. Once a descent direction has been found with the method of choice, a step size α_k should be computed that determines how far $\boldsymbol{m}_k$ should move along that direction to produce the updated model $\boldsymbol{m}_{k+1}$. We recall that the updated model is given by (F-1). In this section, we discuss how the step size α_k can be chosen.

Newton's method has a natural step size of 1. When the objective function being minimised is quadratic, this is exact. However, when it comes to Quasi-Newton methods, Gauss-Newtons methods and others, a step size of 1 may be too large or too small. Therefore, a *line search* should be carried out in the direction $\mathbf{d}_k$ to find the optimal step length α_k. Ideally, we want to find an α_k such that

$$\alpha_k = \underset{\alpha>0}{\operatorname{argmin}}\, \phi(\boldsymbol{m}_k + \alpha \mathbf{d}_k) = \underset{\alpha>0}{\operatorname{argmin}}\, f(\alpha), \tag{F-22}$$

and then we set $\boldsymbol{m}_{k+1} = \boldsymbol{m}_k + \alpha_k \mathbf{d}_k$. Solving (F-22) exactly is called an exact line search. Line search algorithms often do not solve (F-22) exactly, and instead find an α_k that satisfies certain conditions which guarantee that the step size isn't too big or too small. These are called inexact line searches. Popular conditions to impose for the line search are the Wolfe conditions, made up of the Armijo and the curvature conditions. The Armijo condition is

$$\phi(\boldsymbol{m}_k + \alpha_k \mathbf{d}_k) \leq \phi(\boldsymbol{m}_k) + c_1 \alpha_k \nabla \phi(\boldsymbol{m}_k)^T \mathbf{d}_k \tag{F-23}$$

and the curvature condition is

$$\mathbf{d}_k^T \nabla \phi(\boldsymbol{m}_k + \alpha_k \mathbf{d}_k) \geq c_2 \mathbf{d}_k^T \nabla \phi(\boldsymbol{m}_k) \tag{F-24}$$

where $0 < c_1 < c_2 < 1$. The Armijo condition ensures the cost function has sufficiently decreased, and the curvature condition ensures that the slope has been decreased sufficiently.

Remark F-2. *The Wolfe conditions ensure that*

$$\boldsymbol{y}_k^T \boldsymbol{s}_k > 0.$$

An important property of the BFGS update formula (F-9) is that B_{k+1} inherits the positive-definiteness of B_k when $\boldsymbol{y}_k^T \boldsymbol{s}_k > 0$.

A step length may satisfy the Wolfe conditions without being particularly close to the minimiser of f. Alternatively, the Strong Wolfe conditions may be used, which replaces (F-24) with

$$|\mathbf{d}_k^T \nabla \phi(\boldsymbol{m}_k + \alpha_k \mathbf{d}_k)| \leq c_2 |\mathbf{d}_k^T \nabla \phi(\boldsymbol{m}_k)| \tag{F-25}$$

Using this condition, we no longer allow the derivative $f'(\alpha_k)$ to be too positive, and hence ensure that α_k lies close to a critical point of f. A small value of c_2 implies an accurate minimisation. However it may not be computationally efficient to perform an accurate minimization during the line search, and in general, the weaker curvature condition is often adequate for implementation.

The algorithm for the Weak Wolfe line search is given in Algorithm F-3.

Algorithm F-3 Weak Wolfe Line Search

Inputs: $\boldsymbol{m}_k, \phi(\boldsymbol{m}_k), \nabla\phi(\boldsymbol{m}_k), \mathbf{d}_k, c_1, c_2$
Set $\alpha = 1$, $\mu = 0$, $\nu = 0$
Repeat
    if $\phi(\boldsymbol{m}_k + \alpha_k \mathbf{d}_k) > \phi(\boldsymbol{m}_k) + c_1 \alpha \nabla\phi(\boldsymbol{m}_k)^T \mathbf{d}_k$ **then**
        $\nu = \alpha$
        $\alpha = \frac{1}{2}(\mu + \nu)$
    else if $\mathbf{d}_k^T \nabla\phi(\boldsymbol{m}_k + \alpha_k \mathbf{d}_k) < c_2 \mathbf{d}_k^T \nabla\phi(\boldsymbol{m}_k)$ **then**
        $\mu = \alpha$
        if $\nu = 0$
            $\alpha = 2\mu$
        else
            $\alpha = \frac{1}{2}(\mu + \nu)$
        end if
    else
        stop
    end if
end repeat
Output: α

Appendix G

Consistency of the Gauss-Newton Method for FWI

In Section 2.4.3, we discussed the non-regularised FWI problem. We showed that as $\boldsymbol{\varepsilon} \to \mathbf{0}$, $||H^{(2)}||_2 \to 0$ (Corollary 2.4.13) and hence, $H \to H^{(1)}$ (by Theorem 2.4.7). By the definition of Newton's method and the Gauss-Newton method (detailed in Appendix F and [136, Section 10.2]), this means that Newton's method approaches the Gauss-Newton method as $\boldsymbol{\varepsilon} \to \mathbf{0}$. The assumption made when using the Gauss-Newton method is that the Jacobian is full rank and the Gauss-Newton Hessian positive definite, but this assumption does not always hold for FWI. The FWI Jacobian (2.4.30) is low rank and hence $H^{(1)}$ is singular (shown in Theorem 2.4.6), and therefore the Gauss-Newton method in this case does not have a unique solution.

The Gauss-Newton step for FWI is defined as the solution to the following system,

$$H^{(1)}(\boldsymbol{m}_k, \boldsymbol{p})\mathbf{d}_k = -\nabla\phi(\boldsymbol{m}_k, \boldsymbol{p}) \tag{G-1}$$

where $\mathbf{d}_k$ means the descent direction at the kth iteration, and $\nabla\phi(\boldsymbol{m}_k, \boldsymbol{p})$ is the gradient of the objective function being minimised. Equation (G-1) is solved to find the descent direction, which is then used to find an updated model, given by (F-1). These steps are repeated until we find the model $\boldsymbol{m}$ such that $\nabla\phi(\boldsymbol{m}, \boldsymbol{p}) = 0$.

Since $H^{(1)}$ is singular, we expect that the Gauss-Newton step may not be well-defined. However, in the following theorem, we show that the Gauss-Newton linear system (G-1) is consistent and so a solution exists for the system (in fact there are infinitely many solutions due to the rank-deficiency of $H^{(1)}$).

Specifically, Theorem G-1 shows that a vector in the left nullspace of $H^{(1)}$ (which is the same as the right nullspace since $H^{(1)}$ is symmetric matrix) is orthogonal to the right hand side of the system to be solved (G-1) (i.e., the FWI gradient). The column space (or range) of a matrix is always orthogonal to its left nullspace. This means the right-hand side of (G-1) is in the range of $H^{(1)}$ and so the system (G-1) is consistent.

We note that the following theorem is written in discrete notation.

Theorem G-1. *If* $\widetilde{\boldsymbol{m}} \in ker(H^{(1)}(\boldsymbol{m}, \boldsymbol{p}))$ *then* $\widetilde{\boldsymbol{m}}^* \nabla\phi(\boldsymbol{m}, \boldsymbol{p}) = 0$.

Proof. First we restate the definition of the discrete form of the FWI gradient (2.3.10),

$$\nabla\phi(\boldsymbol{m},\boldsymbol{p}) = -\Re\left\{\sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left[\left(R(\boldsymbol{p})\frac{\partial\mathbf{u}(\boldsymbol{m},s,\omega)}{\partial\boldsymbol{m}}\right)^{*}\boldsymbol{\varepsilon}(\boldsymbol{m},\boldsymbol{p},s,\omega)\right]\right\}, \tag{G-2}$$

where $\dfrac{\partial\mathbf{u}}{\partial\boldsymbol{m}} = \left[\dfrac{\partial\mathbf{u}}{\partial m_1}\ \dfrac{\partial\mathbf{u}}{\partial m_1}\dfrac{\partial\mathbf{u}}{\partial m_2}\ldots\dfrac{\partial\mathbf{u}}{\partial m_M}\right] \in \mathbb{C}^{N\times M}$.

If $\widetilde{\boldsymbol{m}} \in ker(H^{(1)}(\boldsymbol{m},\boldsymbol{p}))$, where $\widetilde{\boldsymbol{m}} \in \mathbb{R}^{M\times 1}$, then

$$\widetilde{\boldsymbol{m}}^{*}H^{(1)}\widetilde{\boldsymbol{m}} = 0,$$

where we have dropped the dependencies on $\boldsymbol{m},\boldsymbol{p},s$ and ω for simplicity. Therefore, using (2.4.36) and (2.4.31)

$$\begin{aligned}
0 &= \widetilde{\boldsymbol{m}}^{*}H^{(1)}\widetilde{\boldsymbol{m}}\\
&= \sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\widetilde{\boldsymbol{m}}^{*}\Re(J^{*}J)\widetilde{\boldsymbol{m}} = \sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\widetilde{\boldsymbol{m}}^{*}J^{*}J\widetilde{\boldsymbol{m}}\\
&= \sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\widetilde{\boldsymbol{m}}^{*}\left(R\frac{\partial\mathbf{u}}{\partial\boldsymbol{m}}\right)^{*}\left(R\frac{\partial\mathbf{u}}{\partial\boldsymbol{m}}\right)\widetilde{\boldsymbol{m}}\\
&= \sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left(R\frac{\partial\mathbf{u}}{\partial\boldsymbol{m}}\widetilde{\boldsymbol{m}}\right)^{*}\left(R\frac{\partial\mathbf{u}}{\partial\boldsymbol{m}}\widetilde{\boldsymbol{m}}\right)\\
&= \sum_{s\in\mathcal{S}}\sum_{\omega\in\mathcal{W}}\left\|R\frac{\partial\mathbf{u}}{\partial\boldsymbol{m}}\widetilde{\boldsymbol{m}}\right\|_2^2.
\end{aligned}$$

Since the sum of squares is zero, then we must have, for every source and every frequency,

$$\left\|R\frac{\partial\mathbf{u}}{\partial\boldsymbol{m}}\widetilde{\boldsymbol{m}}\right\|_2 = 0$$

and so,

$$R\frac{\partial\mathbf{u}}{\partial\boldsymbol{m}}\widetilde{\boldsymbol{m}} = \mathbf{0}.$$

Therefore the inner product of the above with the residual $\boldsymbol{\varepsilon}$ is, for each source and frequency,

$$\widetilde{\boldsymbol{m}}^{*}\left(R\frac{\partial\mathbf{u}}{\partial\boldsymbol{m}}\right)^{*}\boldsymbol{\varepsilon} = 0$$

By the definition of the gradient (G-2) and the fact that $\widetilde{\boldsymbol{m}}$ is real, this means that

$$\widetilde{\boldsymbol{m}}^{*}\nabla\phi = 0.$$

■

In conclusion, this theorem shows that the Gauss-Newton method (G-1) is consistent, and hence that the Newton method is consistent as $\boldsymbol{\varepsilon} \to 0$, $H \to H^{(1)}$. A solution to the Newton system will therefore always exist, even when $\boldsymbol{\varepsilon} \to 0$. Although, we should recall that $H^{(1)}$ is rank-deficient and therefore we cannot invert it, we should be able to solve the system stably using another method, such as the conjugate gradient method, for example see [129].

Appendix H

Convexity and Uniqueness

The following provides a summary of important definitions on convex functions, and results on what the degree of convexity of a function can tell us about its minima - in particular whether they exist and are unique. We state definitions and lemmas/theorems for a general function $f \in \Omega$, and these results are then applied in Chapter 2 when discussing the convexity of the FWI objective function and conditions under which we have a unique FWI solution, and in Chapter 3 when investigating the uniqueness of the solution to the sensor placement problem.

Definition H-1. ***Convex Set:*** *A set Ω is convex if, for any $x, y \in \Omega$, and $0 \leq \theta \leq 1$,*

$$\theta x + (1-\theta)y \ \in \Omega.$$

Definition H-2. ***Strictly Convex Function:*** *A function is* ***strictly convex*** *if, for all $x, y \in \Omega$, $x \neq y$ for $0 < \theta < 1$,*

$$f(\theta x + (1-\theta)y) < \theta f(x) + (1-\theta)f(y). \tag{H-1}$$

Note that a **convex** function is defined by (H-1) with the strict inequality replaced by $\leq$, for $0 \leq \theta \leq 1$, and x can equal y.

Lemma H-3. ***Second-Order Condition for Strict Convexity:*** *Suppose a function f is twice differentiable. Then f is strictly convex on Ω if, for all $x \in \Omega$, the Hessian is positive definite,*

$$\nabla_x^2 f(x) \succ 0. \tag{H-2}$$

This lemma is stating that if the Hessian of a function is positive definite everywhere,

then the function is strictly convex. (This can be interpreted geometrically as the function having positive curvature everywhere.) See [25, Proposition 1.1.10] for a proof of Lemma H-3.

Definition H-4. ***Strongly Convex Function:*** *A function is γ-**strongly convex** if, for $\gamma > 0$, and for all $x, y \in \Omega$, for $0 \leq \theta \leq 1$,*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y) - \frac{\gamma}{2}\theta(1-\theta)||x-y||^2. \tag{H-3}$$

Another way of defining strong convexity is to say that f is γ-strongly convex if $f(x) - \frac{\gamma}{2}||x||^2$ is convex for some $\gamma > 0$. A proof of this can be found in [93, Proposition 1.1.2]. We note that strong convexity implies strict convexity.

Lemma H-5. ***Second-Order Condition for Strong Convexity:*** *Suppose a function f is twice differentiable. Then f is γ-strongly convex on Ω if and only if, for $\gamma > 0$, and for all $x \in \Omega$,*

$$\nabla_x^2 f(x) \succeq \gamma I. \tag{H-4}$$

This lemma means that if the smallest eigenvalue of the Hessian of a function is uniformly lower bounded by γ everywhere, then that function is strongly convex. A proof of this can be found in [93, Theorem 4.3.1].

We now look at the implications of strict and strong convexity on the nature of the minima.

Theorem H-6. ***Existence and Uniqueness of Optimal Solutions:***
Consider the optimisation problem

$$\min f(x) \qquad \text{subject to} \quad \text{x} \in \Omega$$

(i) If $f : \mathbb{R}^n \to \mathbb{R}$ is strictly convex on Ω, and Ω is a convex set, then the optimal solution (assuming it exists) is unique.

(ii) If $f : \mathbb{R}^n \to \mathbb{R}$ is strongly convex on Ω, and Ω is a convex set, then the optimal solution exists and is unique.

Proof. (i) The proof of part (i) is based on [25, Proposition 3.1.1] and [179, Proposition

2]. To prove by contradiction, suppose there are two optimal solutions, x and $y \in \mathbb{R}^n$, such that $f(x) = f(y) = f^*$ and $x \neq y$. This means that $x, y \in \Omega$, and

$$f^* = f(x) = f(y) < f(z), \ \ \forall z \in \Omega \tag{H-5}$$

Now consider $z = \alpha x + (1-\alpha)y$, where $\alpha \in (0,1)$. By convexity of Ω, we have $z \in \Omega$. By the definition of strict convexity in (H-1),

$$f(z) = f\left(\alpha x + (1-\alpha)y\right) < \alpha f(x) + (1-\alpha)f(y) = \alpha f^* + (1-\alpha)f^* = f^*,$$

which contradicts (H-5). Therefore the solution must be unique.

(ii) We refer to [19, Corollary 11.16] for a proof of part (ii). ∎

This theorem proves that if a minimum of a strictly convex function exists, it is unique, but there is no guarantee that a minimum of a strictly convex function exists. To guarantee existence, we would need the objective function to be strongly convex. To understand the difference between a strictly convex function and a strongly convex function, the following 1D example is useful to consider. The function $f(x) = e^x$ is strictly convex because $f''(x) > 0$ for all x, but no minimum exists. The function $g(x) = \frac{1}{2}x^2$ is strongly convex with $\gamma = 1$ because $g''(x) = 1$.

Appendix I

Bilevel Optimisation Overview

The parameter optimisation problem that we have formulated in this thesis falls into the framework of bilevel optimisation. Here we provide an overview of the theory of bilevel optimisation, a review of possible solution approaches to the problem, and a list of example applications. The review is based on material from [47], [181], [163], [61], and [60].

History

Firstly, we briefly review the history of bilevel (and multilevel) optimisation problems and their evolution over time.

From a historical point of view, the origin of bilevel optimisation can be traced to the field of economics. Specifically, bilevel optimisation problems were first formulated by Stackelberg [171] in the context of game theory. The strategic game described by Stackelberg, the so-called Stackelberg game, involves a hierarchical leader-follower structure, with is asymmetric in nature. In the game, the players compete with each other, such that the leader makes the first move, and the follower reacts optimally to the leaders move. The leader is aware that the follower observes its actions before reacting optimally. The leader must anticipate the optimal response of the followers to choose their optimal strategy accordingly. Therefore, the leader's optimisation problem contains a nested optimisation problem that corresponds to the followers optimisation problem. This is the structure of a bilevel optimisation problem, where the leaders problem is the upper-level problem, and the followers problem is the lower-level problem.

About 40 years after the Stackelberg game was originally published, problems with this hierarchical structure were introduced into the into the mathematical community. This began in the area of operations research and mathematical programming, where the bilevel optimisation problems were written as an outer optimisation problem with a nested inner optimisation problem appearing as a constraint. These problems were initially considered by Bracken and McGill in [30], with subsequent publications that deal with applications of these problems, both in military and defence [31] and in marketing decision making [29]. At the time of these publications, these problems were termed 'mathematical programs with optimisation problems in the constraints', with the term 'multilevel programming' being introduced in [41].

In the 1980's, the usefulness of bilevel optimisation in engineering design problems

and hierarchical design processes became apparent to researchers [47]. The first literature study on 'bilevel mathematical programming' was published in 1985 [107]. Since then, interest in bilevel optimisation has been continually growing, due to it's number of applications in different fields and its interesting mathematics. A substantial body of literature has been published, including theoretical and numerical investigations, as well as real-life applications of the problem. A bibliography of many important references in this field can be found in [181].

General Formulation

In this section, we provide a general formulation for bilevel optimisation. As previously described, the bilevel problem is made up of two levels of optimisation, the upper-level optimisation problem and the lower-level optimisation problem. Correspondingly, there are two kinds of variables - the upper-level variables x_u and lower-level variables x_l. The nested structure of the problem means that the upper-level problem usually has full knowledge of the lower-level problem, but the lower-level problem only knows the outcomes of the upper-level, and then optimises itself based on this . This means that the upper-level variables are treated as parameters during the lower-level optimisation with respect to x_l. The nested structure of the problem puts a constraint that only the optimal solutions to the lower-level optimisation task may be acceptable as possible feasible candidates to the upper-level optimisation task, i.e., a pair (x_u, x_l^*), where x_l^* is the optimal response to x_u, is defined as a *feasible solution* to the upper-level problem (as long as it also satisfies the constraints of the problem). We formally define a general bilevel problem in Defintion I-1 (adapted from [163] and [164]).

Definition I-1. ***Bilevel Optimisation Problem:*** *For the upper-level objective function $F : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, and lower-level objective function $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$, the bilevel problem is given by*

$$\begin{aligned}
&\min_{x_u \in X_U,\, x_l \in X_L} F(x_u, x_l) \\
&\textit{subject to} \\
&x_l \in \underset{x_l \in X_L}{argmin}\{f(x_u, x_l) : g_j(x_u, x_l) \leq 0, j = 1, ..., J\} \\
&G_k(x_u, x_l) \leq 0 \quad k = 1, .., K
\end{aligned}$$

where $x_u \in X_U \subseteq \mathbb{R}^n$, $x_l \in X_L \subseteq \mathbb{R}^m$ are the upper- and lower-level variables respectively, and $G_k : X_U \times X_L \to \mathbb{R}$, $k = 1, ..., K$ and $g_j : X_U \times X_L \to \mathbb{R}$, $j = 1, ..., J$ are the upper- and lower-level constraints respectively.

In the case where there is a unique lower-level optimal solution, Definition I-1 is well-defined. An ambiguity arises in this problem definition, in the case where there is

more than one optimal lower-level solution for any given upper-level parameter. This ambiguity is handled by assuming one of two positions, optimistic or pessimistic. In an optimistic position, a solution is chosen from the lower-level optimal set which is most favourable to the upper-level, i.e., optimising according to the best case scenario. In a pessimistic position, the opposite approach is taken, and the upper-level optimises its problem according to the worst case scenario.

Solution Approaches

This section provides an overview of popular solution approaches to the bilevel problem.

The conventional first step in solving a bilevel problem is to transform it to a single level. When the lower-level problem is convex and sufficiently regular [163], the lower-level optimisation problem may be replaced by its Karush-Kuhn-Tucker (KKT) conditions, which are a set of equations and inequalities that determine the optimal solutions of an optimisation problem. Replacing the lower-level with its KKT conditions reduces the bilevel problem into a single-level constrained optimisation problem. For example, the problem in Definition I-1 would be yield the following single-level reformulation, assuming convexity and regularity conditions are met,

$$
\begin{aligned}
&\min_{x_u \in X_U,\, x_l \in X_L,\, \lambda} F(x_u, x_l) \\
&\text{subject to} \\
&G_k(x_u, x_l) \leq 0 \quad k = 1, .., K \\
&\nabla_{x_l} L(x_u, x_l, \lambda) = 0, \\
&g_j(x_u, x_l) \leq 0, \;\; j = 1, ..., J \\
&\lambda_j g_j(x_u, x_l) = 0, \;\; j = 1, ..., J \\
&\lambda_j \geq 0, \;\; j = 1, ..., J
\end{aligned}
$$

where

$$
L(x_u, x_l, \lambda) = f(x_u, x_l) + \sum_{j=1}^{J} \lambda_j g_j(x_u, x_l)
$$

is the Lagrangian function associated with the lower-level problem. Many popular solution approaches to bilevel optimisation are based on solving this reduced problem. For example, Branch and Bound approaches have been applied successfully to single-level reductions of the bilevel problem in [15] and [65].

Penalty function methods are a class of algorithms used in constrained optimisation, which belong to some of the earliest solution methods applied to bilevel optimisation problems ([6], [7]). In general, penalty methods work by replacing a constrained optimisation problem by a series of unconstrained problems. The unconstrained problems are formed by adding a penalty function to the objective function. The first applications of the penalty method to bilevel problems involved replacing the lower-level problem with a penalised problem. The drawback of this approach was that the bilevel structure of the

problem was preserved and the resulting penalised problem was not significantly easier to solve than the original bilevel problem. A double penalty method was proposed in [97], where both upper and lower-level objective functions are penalised. The penalised lower-level is then replaced by its stationarity condition, and the problem is reduced to a single level.

Descent methods are another popular approach for solving bilevel optimisation problems. On every iteration of a descent method, a descent direction should be found such that the upper-level is decreased, and the new point found is feasible. Keeping the new point feasible at every iteration means that it should always be lower-level optimal. Therefore, it can be quite challenging to find a feasible descent direction for the general bilevel problem, and assumptions are often made in its derivation. If it is assumed that, for any x_u, there is a unique optimal solution of the lower-level problem, x_l^*, and that this is an implicit function of x_u, i.e., $x_l^*(x_u)$, then the bilevel problem may be viewed in terms of upper-level variables x_u [47]. Given a feasible point x_u, it is aimed to find the descent direction that results in a sufficient decrease of the upper-level F, while maintaining feasibility of the problem. This can be achieved using gradient information, which requires the computation of the gradient of the upper-level objective function $\nabla_{x_u} F(x_u, x_l^*(x_u))$ at a feasible point. Assuming $\nabla_{x_u} x_l^*$ is well-defined, applying the chain rule gives the following expression for the gradient, evaluated at a feasible iterate $(x_u, x_l^*(x_u))$,

$$\nabla_{x_u} F(x_u, x_l^*(x_u)) = \nabla_{x_u} F(x_u, x_l^*(x_u)) + \nabla_{x_l} F(x_u, x_l^*(x_u)) \nabla_{x_u} x_l^*(x_u). \tag{I-1}$$

To find an expression for $\nabla_{x_u} x_l^*$, we can use the fact that the first-order optimality conditions should be satisfied for the lower-level, i.e., $\nabla_{x_l} f(x_u, x_l^*(x_u)) = 0$. We also make assumptions that x_l^* is continuously differentiable at x_u, that the lower-level is twice-continuously differentiable and that the Hessian of the lower-level at the feasible points is invertible ([138]). Under these assumptions, it holds that

$$\frac{dx_l^*}{dx_u}(x_u) = -\left(\frac{\partial^2 f}{\partial x_l^2}(x_u, x_l^*(x_u))\right)^{-1} \frac{\partial f}{\partial x_u \partial x_l}(x_u, x_l^*(x_u)). \tag{I-2}$$

The computation of this gradient can be expensive. It also relies on many assumptions. For non-smooth lower-level problems that do not nexessarily have a unique solution, [138] proposes techniques for approximating bilevel optimisation problems.

Trust-region methods have also been applied in solving bilevel problems. Trust-region methods involve approximating a region of the objective function with a so-called model function. If this approximation is good, the region is expanded, and if not, the region is contracted. Trust region methods have been applied successfully to bilevel optimisation problems in [116] and [46], for example. More details of trust-region algorithms in bilevel optimisation can be found in [60].

Due to the difficult nature of bilevel problems, many of the solution techniques discussed have involved simplifying assumptions of, for example, smoothness and convexity. Sometimes these classical approaches may fail due to real-world difficulties, and so research into solution methods for more complex bilevel problems is ongoing, for example recent studies on evolutionary algorithms (as reviewed in [163]), and meta-modelling (see [105]).

Applications

Bilevel optimisation frequently arises in practical problems. The research into bilevel problems has been strongly motivated by these real-world applications. Here we will briefly discuss some of these applications.

Bilevel optimisation problems are often applied in the chemical industry. When producing substances through chemical reactions, chemists and engineers have to decide on the conditions of the reaction (for example temperature of reactor and quantities of reactants) that will result in the correct substance being produced, and for this output to be optimal. An optimal output in this case would be defined as producing as large an amount as possible of the required substance, and the amount of unwanted, or dangerous, by-products to be as small as possible. The upper-level of this problem involves optimising the output of the reaction, and the lower-level is an energy minimisation problem involving a chemical equilibria equation (this ensures the correct substance is produced). Applications of bilevel optimisation in this area includes [44], [87] and [153].

Bilevel optimisation is also applied to problems in optimal design. For example, the design of structures can be formulated as a bilevel optimisation problem with constraints, in order to choose the amount and types of materials for the structure, the shape of the structure, etc. The upper-level optimisation task often requires the minimisation of the weight or cost of the structure, with constraints involving displacements, contact forces and stresses. The lower-level is a potential energy minimisation problem. More information on optimal design can be found in [106], [90], and [43].

As mentioned previously, the bilevel problem with applications in military and defence has been formulated in [31]. An example of an application in defence includes offensive and defensive strategy design. When designing an offensive strategy, the 'leader' can be seen as the offensive entity and the 'follower' as the defensive entity (using the terminology of the Stackelberg game). The offensive entity wants to maximise the damage caused to its opponent, but it can only do this optimally if it takes into account the reactions of the defensive entity. The defensive entity always wants to react optimally to the attack. Therefore, the offensive entity's optimisation problem is the upper-level task, and the lower-level optimisation problem involves computing the optimal response of the defensive entity to the offensive entity's actions. Conversely, when designing a defensive strategy, the defensive entity is the leader and the offensive entity is the follower. At the lower-level, the attacker maximises the damage it causes, while at the upper-level, the defence aims to choose optimal strategies that will minimise this damage. Some other recent applications in this area can be found in [9], [37] and [36].

Applications of bilvel optimisation to water-management can be found in [12], [27] and [5]. For example, in [12], multilevel optimisation is applied to international river management in India and Bangladesh. These countries share water from the Ganges river, and there are a series of dams in both countries which they use for hydroelectric power, irrigation and flood protection. This paper presents the problem of coordinating resources in the form of a Stackelberg game, where the multilevel problem was investigated in the different cases where India, Bangladesh or an arbitrator (the

UN) was the leader.

This section has provided some examples of where bilevel, and multilevel, optimisation has been applied in the past. A wide-range of other applications also exist which haven't been covered in this section, for example, revenue management [51], facility location [108], and minimising greenhouse gases [91].

Learning in Bilevel Optimisation

An important use of bilevel optimisation is in the approach to learning. In this section, we will focus on applications of bilevel learning in imaging. Applications include parameter learning in image restoration and denoising, for example [109], [58] and [59], or learning a sampling pattern for MRI imaging [162]. Learning problems require knowledge of the problem in terms of a training set. Based on this prior knowledge, optimal parameters can be learned in what is known as a supervised learning method.

These type of learning problems are formulated with a bilevel optimisation approach, where the lower-level is an imaging reconstruction problem. Using the notation in [59], the general form of the lower-level objective function is

$$\alpha R(u) + d(K(u), f)$$

where R is a regularisation term, α is the regularisation parameter, f is some given data which is related to an image u through a forward operator, or function, K and d is a suitable distance function. The aim of the lower-level image reconstruction problem is to find the image u from data f. The lower-level has a number of aspects which can be learned, for example the regularisation parameter α, the type of regularisation term R, the type of fidelity term d, or, if applicable, what to measure and where to take measurements. Different choices lead to different reconstructions. The upper-level objective function is then a loss function, that measures the difference between the solution of the lower-level and the training set. Through minimisation of the upper-level objective function, the optimal parameters are learned. The training set can take the form of pairs of clean and noisy images (see [58] and [59]), or pairs of clean images and corresponding noisy measured data (see for example [162]), depending on the problem. As an example, the general bilevel learning problem is written in [59] as

$$\begin{aligned} &\min F(u^*) = cost(u^*, f_0) \\ \text{subject to} \quad &u^* \in \underset{u}{\operatorname{argmin}}\{\alpha R(u) + d(K(u), f)\} \end{aligned}$$

where F is a cost functional, and the training set is (f, f_0) (noisy and clean images respectively). The argument of the upper-level minimisation problem depends on the parameters that are being learned.

In the area of geophysical imaging, bilevel learning has been applied in [85]. Here the lower-level is traveltime tomography for reconstructing the subsurface, and on the upper-level a regularisation functional is learned.

Appendix J

Implicit Function Theorem

Consider the system of m equations

$$f_i(y_1, ..., y_m, x_1, ..., x_n) = 0, \qquad i = 1, ..., m,$$

which can be abbreviated to

$$\mathbf{f}(\mathbf{y}, \mathbf{x}) = \mathbf{0}.$$

The implicit function theorem states that, under a condition on the partial derivatives with respect to the y_i's, at a point, the y_i variables are differentiable functions of the x_j variables in some neighbourhood of that point. The formal statement of the implicit function theorem is below.

Theorem J-1. ***Implicit Function Theorem:*** *Let $\boldsymbol{f}: \mathbb{R}^{m+n} \to \mathbb{R}^m$ be a continuously differentiable function. Let $\mathbb{R}^{m+n}$ have coordinates $(\boldsymbol{y}, \boldsymbol{x})$ with $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{x} \in \mathbb{R}^n$. The Jacobian matrix with respect to $\boldsymbol{y}$ is defined as*

$$J_{i,j}(\boldsymbol{y}, \boldsymbol{x}) = \left[\frac{\partial f_i}{\partial y_j}(\boldsymbol{y}, \boldsymbol{x})\right], \qquad \text{for} \quad i, j = 1, ..m.$$

Let $(\boldsymbol{b}, \boldsymbol{a})$ be a point such that $\boldsymbol{f}(\boldsymbol{b}, \boldsymbol{a}) = \boldsymbol{0}$. Assuming the Jacobian evaluated at $(\boldsymbol{b}, \boldsymbol{a})$ is invertible, i.e., $\det J(\boldsymbol{b}, \boldsymbol{a}) \neq 0$, then there exists an open set $U \subset \mathbb{R}^n$ containing $\boldsymbol{a}$ and a unique continuously differentiable function $\boldsymbol{g}: U \to \mathbb{R}^m$ such that $g(\boldsymbol{a}) = \boldsymbol{b}$ and $\boldsymbol{f}(\boldsymbol{g}(\boldsymbol{x}), \boldsymbol{x}) = \boldsymbol{0}$ for all $\boldsymbol{x} \in U$.
In addition, the partial derivatives of $\boldsymbol{g}$ in U are given by the matrix-vector product

$$\left[\frac{\partial \boldsymbol{g}}{\partial x_j}(\boldsymbol{x})\right]_{m\times 1} = -\left[J(\boldsymbol{g}(\boldsymbol{x}), \boldsymbol{x})\right]^{-1}_{m\times m} \left[\frac{\partial \boldsymbol{f}}{\partial x_j}(\boldsymbol{g}(\boldsymbol{x}), \boldsymbol{x})\right]_{m\times 1}.$$

Remark J-2. *If $\boldsymbol{f}$ is continuously differentiable k times then the same holds true for g inside U [89, Theorem 9.2.3]. Furthermore, if $\boldsymbol{f} \in C^\infty$, then $g \in C^\infty$ in U [170, page 186].*

Bibliography

[1] *How does the oil and gas industry discover new reserves in offshore locations?* https://oilfieldteam.com/en/a/learning/discover-new-reserves. Accessed: 30-12-2021.

[2] *Innoseis Seismic Surveying.* http://www.innoseis.com/seismic-surveying. Accessed: 30-12-2021.

[3] *Polarcus Amani Ulstein SX134.* https://www.firegrader.com/portfolio/polarcus-amani-ulstein/. Accessed: 30-12-2021.

[4] *The Keys to High-Potential Exploration.* https://www.ep.total.com/en/expertise/exploration/keys-high-potential-exploration. Accessed:30-12-2021.

[5] I. Ahmad, F. Zhang, J. Liu, M. N. Anjum, M. Zaman, M. Tayyab, M. Waseem, and H. U. Farid, *A linear bi-level multi-objective program for optimal allocation of water resources*, PLoS One, 13 (2018), p. e0192294.

[6] E. Aiyoshi and K. Shimizu, *Hierarchical decentralized systems and its new solution by a barrier method.*, IEEE Transactions on Systems, Man and Cybernetics, (1981), pp. 444–449.

[7] ——, *A solution method for the static constrained stackelberg problem via penalty method*, IEEE Transactions on Automatic Control, 29 (1984), pp. 1111–1114.

[8] K. Aki and P. G. Richards, *Quantitative Seismology*, 2002.

[9] D. Aksen and N. Aras, *A bilevel fixed charge location model for facilities under imminent attack*, Computers & Operations Research, 39 (2012), pp. 1364–1381.

[10] G. Alessandrini and R. Gaburro, *The local Calderon problem and the determination at the boundary of the conductivity*, Communications in Partial Differential Equations, 34 (2009), pp. 918–936.

[11] G. Alessandrini, V. Maarten, F. Faucher, R. Gaburro, and E. Sincich, *Inverse problem for the Helmholtz equation with Cauchy data: Reconstruction with conditional well-posedness driven iterative regularization*, ESAIM: Mathematical Modelling and Numerical Analysis, 53 (2019), pp. 1005–1030.

[12] G. Anandalingam and V. Apprey, *Multi-level programming and conflict resolution*, European Journal of Operational Research, 51 (1991), pp. 233–247.

[13] C. Ashton, B. Bacon, A. Mann, N. Moldoveanu, C. Déplanté, D. Ireson, T. Sinclair, and G. Redekop, *3D seismic survey design*, Oilfield Review;(Netherlands), 6 (1994).

[14] A. Asnaashari, R. Brossier, S. Garambois, F. Audebert, P. Thore, and J. Virieux, *Regularized seismic full waveform inversion with prior model information*, Geophysics, 78 (2013), pp. R25–R36.

[15] J. F. Bard and J. T. Moore, *A branch and bound algorithm for the bilevel programming problem*, SIAM Journal on Scientific and Statistical Computing, 11 (1990), pp. 281–292.

[16] N. Barth and C. Wunsch, *Oceanographic experiment design by simulated annealing*, Journal of Physical Oceanography, 20 (1990), pp. 1249–1263.

[17] H. Barucq, G. Chavent, and F. Faucher, *A priori estimates of attraction basins for nonlinear least squares, with application to Helmholtz seismic inverse problem*, Inverse Problems, 35 (2019), p. 115004.

[18] C. C. Bates, T. F. Gaskell, and R. B. Rice, *Geophysics in the affairs of man: A personalized history of exploration geophysics and its allied sciences of seismology and oceanography*, Elsevier, 2013.

[19] H. H. Bauschke, P. L. Combettes, et al., *Convex analysis and monotone operator theory in Hilbert spaces*, vol. 408, Springer, 2011.

[20] A. Bayliss, M. Gunzburger, and E. Turkel, *Boundary conditions for the numerical solution of elliptic equations in exterior regions*, SIAM Journal on Applied Mathematics, 42 (1982), pp. 430–451.

[21] A. Behzadan and M. Holst, *Multiplication in sobolev spaces, revisited*, arXiv preprint arXiv:1512.07379, (2015).

[22] M. Bellassoued, *Carleman estimates and distribution of resonances for the transparent obstacle and application to the stabilization*, Asymptotic Analysis, 35 (2003), pp. 257–279.

[23] A. Ben-Menahem, *A concise history of mainstream seismology: Origins, legacy, and perspectives*, Bulletin of the Seismological Society of America, 85 (1995), pp. 1202–1225.

[24] E. Beretta, M. V. De Hoop, F. Faucher, and O. Scherzer, *Inverse boundary value problem for the Helmholtz equation: quantitative conditional Lipschitz stability estimates*, SIAM Journal on Mathematical Analysis, 48 (2016), pp. 3962–3983.

[25] D. P. Bertsekas, *Convex optimization theory*, Athena Scientific Belmont, 2009.

[26] C. M. Bishop, *Pattern recognition and Machine Learning*, Springer, 2006.

[27] J. Bisschop, W. Candler, J. H. Duloy, and G. T. O'Mara, *The indus basin model: a special application of two-level linear programming*, in Applications, Springer, 1982, pp. 30–38.

[28] S. Borguet and O. Léonard, *The Fisher information matrix as a relevant tool for sensor selection in engine health monitoring*, International Journal of Rotating Machinery, 2008 (2008).

[29] J. Bracken and J. McGill, *Production and marketing decisions with multiple objectives in a competitive environment*, Journal of Optimization Theory and Applications, 24 (1978), pp. 449–458.

[30] J. Bracken and J. T. McGill, *Mathematical programs with optimization problems in the constraints*, Operations Research, 21 (1973), pp. 37–44.

[31] ——, *Defense applications of mathematical programs with optimization problems in the constraints*, Operations Research, 22 (1974), pp. 1086–1096.

[32] J. H. Bramble and J. E. Pasciak, *A note on the existence and uniqueness of solutions of frequency domain elastic wave problems: A priori estimates in h1*, Journal of Mathematical Analysis and Applications, 345 (2008), pp. 396–404.

[33] S. C. Brenner and L. R. Scott, *The mathematical theory of finite element methods*, vol. 3, Springer, 2008.

[34] R. Brossier, S. Operto, and J. Virieux, *Robust elastic frequency-domain full-waveform inversion using the L1 norm*, Geophysical Research Letters, 36 (2009).

[35] R. Brossier, S. Operto, and J. Virieux, *Seismic imaging of complex onshore structures by 2D elastic frequency-domain full-waveform inversion*, Geophysics, 74 (2009), pp. WCC105–WCC118.

[36] G. Brown, M. Carlyle, D. Diehl, J. Kline, and K. Wood, *A two-sided optimization for theater ballistic missile defense*, Operations research, 53 (2005), pp. 745–763.

[37] G. G. Brown, W. M. Carlyle, R. C. Harney, E. M. Skroch, and R. K. Wood, *Interdicting a nuclear-weapons project*, Operations Research, 57 (2009), pp. 866–877.

[38] C. Bunks, F. M. Saleck, S. Zaleski, and G. Chavent, *Multiscale seismic waveform inversion*, Geophysics, 60 (1995), pp. 1457–1473.

[39] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, *A limited memory algorithm for bound constrained optimization*, SIAM Journal on scientific computing, 16 (1995), pp. 1190–1208.

[40] A. Cai and C. A. Zelt, *Data weighted full-waveform inversion with adaptive moment estimation for near-surface seismic refraction data*, in SEG International Exposition and Annual Meeting, OnePetro, 2019.

[41] W. Candler and R. Norton, *Multi-level programming and development policy*, The World Bank, 1977.

[42] C. Castellanos, L. Métivier, S. Operto, R. Brossier, and J. Virieux, *Fast full waveform inversion with source encoding and second-order optimization methods*, Geophysical Journal International, 200 (2015), pp. 720–744.

[43] S. Christiansen, M. Patriksson, and L. Wynter, *Stochastic bilevel programming in structural optimization*, Structural and multidisciplinary optimization, 21 (2001), pp. 361–371.

[44] P. A. Clark and A. W. Westerberg, *Bilevel programming for steady-state chemical process design — I. Fundamentals and algorithms*, Computers & Chemical Engineering, 14 (1990), pp. 87–97.

[45] D. A. Coles and F. D. Morgan, *A method of fast, sequential experimental design for linearized geophysical inverse problems*, Geophysical Journal International, 178 (2009), pp. 145–158.

[46] B. Colson, P. Marcotte, and G. Savard, *A trust-region method for nonlinear bilevel programming: algorithm and computational experience*, Computational Optimization and Applications, 30 (2005), pp. 211–227.

[47] ——, *An overview of bilevel optimization*, Annals of operations research, 153 (2007), pp. 235–256.

[48] D. Colton and R. Kress, *Inverse acoustic and electromagnetic scattering theory*, vol. 93, Springer Nature, 2019.

[49] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to derivative-free optimization*, SIAM, 2009.

[50] J. K. Cooper, G. F. Margrave, and D. C. Lawton, *Simulations of seismic acquisition footprint*, CREWES Res. Rep., 19 (2007), pp. 1–28.

[51] J. P. Côté, P. Marcotte, and G. Savard, *A bilevel modelling approach to pricing and fare optimisation in the airline industry*, Journal of Revenue and Pricing Management, 2 (2003), pp. 23–36.

[52] E. Crase, A. Pica, M. Noble, J. McDonald, and A. Tarantola, *Robust elastic nonlinear waveform inversion: Application to real data*, Geophysics, 55 (1990), pp. 527–538.

[53] A. Curtis, A. Michelini, D. Leslie, and A. Lomax, *A deterministic algorithm for experimental design applied to tomographic and microseismic monitoring surveys*, Geophysical Journal International, 157 (2004), pp. 595–606.

[54] F. T. da Costa, M. A. C. Santos, and D. M. Soares Filho, *Wavenumbers illuminated by time-domain acoustic FWI using the L1 and L2 norms*, Journal of Applied Geophysics, 174 (2020), p. 103935.

[55] S. L. E. F. da Silva, P. T. C. Carvalho, C. A. N. da Costa, J. M. de Araújo, and G. Corso, *An objective function for full-waveform inversion based on frequency-dependent offset-preconditioning*, Plos one, 15 (2020), p. e0240999.

[56] W. Dahmen, B. Faermann, I. G. Graham, W. Hackbusch, and S. Sauter, *Inverse inequalities on non-quasi-uniform meshes and application to the mortar element method*, Mathematics of computation, 73 (2004), pp. 1107–1138.

[57] F. W. de Freitas Silva, S. L. E. F. da Silva, M. V. C. Henriques, and G. Corso, *Using fish lateral line sensing to improve seismic acquisition and processing*, Plos one, 14 (2019), p. e0213847.

[58] J. C. De los Reyes and C. B. Schönlieb, *Image denoising: learning the noise model via nonsmooth PDE-constrained optimization*, Inverse Problems & Imaging, 7 (2013), pp. 1183–1214.

[59] J. C. De los Reyes, C.-B. Schönlieb, and T. Valkonen, *Bilevel parameter learning for higher-order total variation regularisation models*, Journal of Mathematical Imaging and Vision, 57 (2017), pp. 1–25.

[60] S. Dempe, *Foundations of bilevel programming*, Springer Science & Business Media, 2002.

[61] ——, *Bilevel optimization: theory, algorithms and applications*, TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik, 2018.

[62] J. E. Dennis and J. J. Moré, *A characterization of superlinear convergence and its application to quasi-newton methods*, Mathematics of computation, 28 (1974), pp. 549–560.

[63] D. I. Dolci, F. A. Silva, P. S. Peixoto, and E. V. Volpe, *Effectiveness and computational efficiency of absorbing boundary conditions for full-waveform inversion*, Geoscientific Model Development Discussions, (2022), pp. 1–36.

[64] S. C. Downing, S. Gazzola, I. G. Graham, and E. A. Spence, *Sensor placement optimisation in seismic inversion via bilevel learning*. In Preparation, 2022.

[65] T. A. Edmunds and J. F. Bard, *Algorithms for nonlinear bilevel mathematical programs*, IEEE transactions on Systems, Man, and Cybernetics, 21 (1991), pp. 83–89.

[66] B. Engquist and A. Majda, *Absorbing boundary conditions for numerical simulation of waves*, Proceedings of the National Academy of Sciences, 74 (1977), pp. 1765–1766.

[67] S. Esterhazy and J. M. Melenk, *On stability of discretizations of the Helmholtz equation*, in Numerical analysis of multiscale problems, Springer, 2012, pp. 285–324.

[68] J. Y. Fan and Y. X. Yuan, *On the quadratic convergence of the levenberg-marquardt method without nonsingularity assumption*, Computing, 74 (2005), pp. 23–39.

[69] F. Faucher, *Contributions to seismic full waveform inversion for time harmonic wave equations: stability estimates, convergence analysis, numerical experiments involving large scale optimization algorithms*, PhD thesis, 2017.

[70] F. Faucher, O. Scherzer, and H. Barucq, *Eigenvector models for solving the seismic inverse problem for the Helmholtz equation*, Geophysical Journal International, 221 (2020), pp. 394–414.

[71] A. Fichtner, *Full seismic waveform modelling and inversion*, Springer Science & Business Media, 2010.

[72] T. C. Frelet, *Finite element approximation of Helmholtz problems with application to seismic wave propagation*, PhD thesis, INSA de Rouen, 2015.

[73] S. I. Friedly and E. Holst, *Land seismic sensor deployment*, June 11 2015. US Patent App. 14/562,953.

[74] E. V. Gallant, R. R. Stewart, D. C. Lawton, M. B. Bertram, and C. Rodriguez, *New technologies in marine seismic surveying: Overview and physical modelling experiments*, 1996.

[75] O. Gauthier, J. Virieux, and A. Tarantola, *Two-dimensional nonlinear inversion of seismic waveforms: Numerical results*, Geophysics, 51 (1986), pp. 1387–1403.

[76] E. J. Gillin, *Seismology's acoustic debt: Robert Mallet, Chladni's figures, and the Victorian science of earthquakes*, Sound Studies, 6 (2020), pp. 65–82.

[77] A. Górszczyk, S. Operto, and M. Malinowski, *Toward a robust workflow for deep crustal imaging by FWI of OBS data: The eastern Nankai Trough revisited*, Journal of Geophysical Research: Solid Earth, 122 (2017), pp. 4601–4630.

[78] I. G. Graham, O. R. Pembery, and E. A. Spence, *The Helmholtz equation in heterogeneous media: a priori bounds, well-posedness, and resonances*, Journal of Differential Equations, 266 (2019), pp. 2869–2923.

[79] I. G. Graham and S. Sauter, *Stability and finite element error analysis for the Helmholtz equation with variable coefficients*, Mathematics of Computation, 89 (2020), pp. 105–138.

[80] B. Granzow, *Bgranzow/l-bfgs-b: A pure MATLAB implementation of L-BFGS-B*. https://github.com/bgranzow/L-BFGS-B.

[81] L. Guasch, O. C. Agudo, M. X. Tang, P. Nachev, and M. Warner, *Full-waveform inversion imaging of the human brain*, NPJ digital medicine, 3 (2020), pp. 1–12.

[82] L. Guasch, M. Warner, and C. Ravaut, *Adaptive waveform inversion: Practice*, Geophysics, 84 (2019), pp. R447–R461.

[83] T. Ha, W. Chung, and C. Shin, *Waveform inversion using a back-propagation algorithm and a huber function norm*, Geophysics, 74 (2009), pp. R15–R24.

[84] E. Haber, L. Horesh, and L. Tenorio, *Numerical methods for experimental design of large-scale linear ill-posed inverse problems*, Inverse Problems, 24 (2008), p. 055012.

[85] E. Haber and L. Tenorio, *Learning regularization functionals—a supervised training approach*, Inverse Problems, 19 (2003), p. 611.

[86] J. Hadamard, *Sur les problèmes aux dérivées partielles et leur signification physique*, Princeton university bulletin, (1902), pp. 49–52.

[87] W. Halter and S. Mostaghim, *Bilevel optimization of multi-component chemical systems using particle swarm optimization*, in 2006 IEEE International Conference on Evolutionary Computation, IEEE, 2006, pp. 1240–1247.

[88] M. Hardt and F. Scherbaum, *The design of optimum networks for aftershock recordings*, Geophysical Journal International, 117 (1994), pp. 716–726.

[89] B. Hasselblatt and A. Katok, *A first course in dynamics: with a panorama of recent developments*, Cambridge University Press, 2003.

[90] J. Herskovits, A. Leontiev, G. Dias, and G. Santos, *Contact shape optimization: a bilevel programming approach*, Structural and multidisciplinary optimization, 20 (2000), pp. 214–221.

[91] G. Hibino, M. Kainuma, Y. Matsuoka, and T. Morita, *Two-level mathematical programming for analyzing subsidy options to reduce greenhouse-gas emissions*, (1996).

[92] B. T. Hinson, *Observability-based guidance and sensor placement*, PhD thesis, 2014.

[93] J.-B. Hiriart-Urruty and C. Lemaréchal, *Fundamentals of convex analysis*, Springer Science & Business Media, 2004.

[94] R. A. Horn and C. R. Johnson, *Matrix analysis*, Cambridge university press, 2012.

[95] Y. Huang, S. A. Ludwig, and F. Deng, *Sensor optimization using a genetic algorithm for structural health monitoring in harsh environments*, Journal of Civil Structural Health Monitoring, 6 (2016), pp. 509–519.

[96] International Association of Geophysical Contractors, *Marine Seismic Operations: An Overview*, 2002.

[97] Y. Ishizuka and E. Aiyoshi, *Double penalty method for bilevel optimization problems*, Annals of Operations Research, 34 (1992), pp. 73–88.

[98] H. Jun, J. Kwon, C. Shin, H. Zhou, and M. Cogan, *Regularized Laplace–Fourier-Domain Full Waveform Inversion Using a Weighted l_2 Objective Function*, Pure and Applied Geophysics, 174 (2017), pp. 955–980.

[99] S. I. Kabanikhin, *Definitions and examples of inverse and ill-posed problems*, Journal of Inverse and Ill-Posed Problems, 16 (2008), pp. 317–357.

[100] J. Kaipio and E. Somersalo, *Statistical and computational inverse problems*, vol. 160, Springer Science & Business Media, 2006.

[101] D. C. Kammer, *Sensor placement for on-orbit modal identification and correlation of large space structures*, Journal of Guidance, Control, and Dynamics, 14 (1991), pp. 251–259.

[102] E. Kase and T. Ross, *Seismic imaging as a means to evaluate foundation structures*, in Current Practices and Future Trends in Deep Foundations, 2004, pp. 361–369.

[103] M. Khodja, M. Prange, and H. Djikpesse, *Guided bayesian optimal experimental design*, Inverse Problems, 26 (2010), p. 055008.

[104] L. Knopoff and A. F. Gangi, *Seismic reciprocity*, Geophysics, 24 (1959), pp. 681–691.

[105] J. Knowles and H. Nakayama, *Meta-modeling in multiobjective optimization*, in Multiobjective optimization, Springer, 2008, pp. 245–284.

[106] M. Kočvara, *Topology optimization with displacement constraints: a bilevel programming approach*, Structural optimization, 14 (1997), pp. 256–263.

[107] C. D. Kolstad, *A review of the literature on bi-level mathematical programming*, tech. rep., Los Alamos National Laboratory Los Alamos, NM, 1985.

[108] H. Küçükaydin, N. Aras, and I. K. Altinel, *Competitive facility location problem with attractiveness adjustment of the follower: A bilevel programming model and its solution*, European Journal of Operational Research, 208 (2011), pp. 206–220.

[109] K. Kunisch and T. Pock, *A bilevel optimization approach for parameter learning in variational models*, SIAM Journal on Imaging Sciences, 6 (2013), pp. 938–983.

[110] D. Lafontaine, E. A. Spence, and J. Wunsch, *For most frequencies, strong trapping has a weak effect in frequency-domain scattering*, Communications on Pure and Applied Mathematics, 74 (2021), pp. 2025–2063.

[111] P. Lailly and J. Bednar, *The seismic inverse problem as a sequence of before stack migrations*, in Conference on inverse scattering: theory and application, SIAM Philadelphia, PA, 1983, pp. 206–220.

[112] W. H. Lee, P. Jennings, C. Kisslinger, and H. Kanamori, *International handbook of earthquake & engineering seismology*, Elsevier, 2002.

[113] E. Lenticchia, R. Ceravolo, and P. Antonaci, *Sensor placement strategies for the seismic monitoring of complex vaulted structures of the modern architectural heritage*, Shock and Vibration, 2018 (2018).

[114] K. Levenberg, *A method for the solution of certain non-linear problems in least squares*, Quarterly of applied mathematics, 2 (1944), pp. 164–168.

[115] D. Li, H. Li, and C. Fritzen, *The connection between effective independence and modal kinetic energy methods for sensor placement*, Journal of sound and vibration, 305 (2007), pp. 945–955.

[116] G. Liu, J. Han, and S. Wang, *A trust region algorithm for bilevel programing problems*, Chinese science bulletin, 43 (1998), pp. 820–824.

[117] S. Liu, J. Geng, and W. Feng, *Controlled illumination and seismic acquisition geometry for target-oriented imaging*, Applied Geophysics, 2 (2005), pp. 230–234.

[118] S. X. Liu, *Symmetry and asymmetry analysis and its implications to computer-aided diagnosis: A review of the literature*, Journal of biomedical informatics, 42 (2009), pp. 1056–1064.

[119] M. Locatelli and F. Schoen, *Global optimization: theory, algorithms, and applications*, SIAM, 2013.

[120] F. Lucka, M. Pérez-Liva, B. E. Treeby, and B. T. Cox, *High resolution 3D ultrasonic breast imaging by time-domain full waveform inversion*, Inverse Problems, 38 (2021), p. 025008.

[121] C. Lyu, Y. Capdeville, D. Al-Attar, and L. Zhao, *Intrinsic non-uniqueness of the acoustic full waveform inverse problem*, Geophysical Journal International, 226 (2021), pp. 795–802.

[122] D. W. Marquardt, *An algorithm for least-squares estimation of nonlinear parameters*, Journal of the society for Industrial and Applied Mathematics, 11 (1963), pp. 431–441.

[123] H. Maurer, A. Curtis, and D. E. Boerner, *Recent advances in optimized geophysical survey design*, Geophysics, 75 (2010), pp. 75A177–75A194.

[124] A. Mazzotti, N. Bienati, E. Stucchi, A. Tognarelli, M. Aleardi, and A. Sajeva, *Two-grid genetic algorithm full-waveform inversion*, The Leading Edge, 35 (2016), pp. 1068–1075.

[125] W. McLean, *Strongly elliptic systems and boundary integral equations*, Cambridge University Press, 2000.

[126] W. Menke, *Geophysical data analysis: Discrete inverse theory*, Academic press, 2018.

[127] L. Métivier, R. Brossier, S. Operto, and J. Virieux, *Second-order adjoint state methods for full waveform inversion*, in EAGE 2012-74th European Association of Geoscientists and Engineers Conference and Exhibition, 2012.

[128] ——, *Full waveform inversion and the truncated newton method*, SIAM review, 59 (2017), pp. 153–195.

[129] L. Métivier, R. Brossier, J. Virieux, and S. Operto, *Full waveform inversion and the truncated newton method*, SIAM Journal on Scientific Computing, 35 (2013), pp. B401–B437.

[130] J. Modersitzki, *FAIR: flexible algorithms for image registration*, SIAM, 2009.

[131] D. Moreno-Salinas, A. Pascoal, and J. Aranda, *Optimal sensor placement for acoustic underwater target positioning with range-only measurements*, IEEE Journal of Oceanic Engineering, 41 (2016), pp. 620–643.

[132] T. P. Nangoo, *Seismic full-waveform inversion of 3D field data: from the near surface to the reservoir*, PhD thesis, Imperial College London, 2014.

[133] P. A. Narbel, J. P. Hansen, and J. R. Lien, *Energy technologies and economics*, Springer, 2014.

[134] F. Nataf, *Absorbing boundary conditions and perfectly matched layers in wave propagation problems*, 2013.

[135] J. Nocedal, A. Sartenaer, and C. Zhu, *On the behavior of the gradient norm in the steepest descent method*, Computational Optimization and Applications, 22 (2002), pp. 5–35.

[136] J. Nocedal and S. Wright, *Numerical optimization*, Springer Science & Business Media, 2006.

[137] O. Novotny, *Seismic surface waves*, Bahia, Salvador: Instituto de Geociencias, 61 (1999).

[138] P. Ochs, R. Ranftl, T. Brox, and T. Pock, *Techniques for gradient-based bilevel optimization with non-smooth lower level problems*, Journal of Mathematical Imaging and Vision, 56 (2016), pp. 175–194.

[139] J. W. Oh and D. J. Min, *Robust scaling strategy for frequency-domain acoustic full waveform inversion*, ASEG Extended Abstracts, 2013 (2013), pp. 1–4.

[140] D. Oldroyd, F. Amador, J. Kozák, A. Carneiro, and M. Pinto, *The study of earthquakes in the hundred years following the lisbon earthquake of 1755*, Earth Sciences History, 26 (2007), pp. 321–370.

[141] S. Operto, J. Virieux, J.-X. Dessa, and G. Pascal, *Crustal seismic imaging from multifold ocean bottom seismometer data by frequency domain full waveform tomography: Application to the eastern Nankai trough*, Journal of Geophysical Research: Solid Earth, 111 (2006).

[142] K. Otto and E. Larsson, *Iterative solution of the Helmholtz equation by a second-order method*, SIAM Journal on Matrix Analysis and Applications, 21 (1999), pp. 209–229.

[143] R. E. Plessix, *A review of the adjoint-state method for computing the gradient of a functional with geophysical applications*, Geophysical Journal International, 167 (2006), pp. 495–503.

[144] R. E. Plessix, G. Baeten, J. W. de Maag, M. Klaassen, Z. Rujie, and T. Zhifei, *Application of acoustic full waveform inversion to a low-frequency large-offset land data set*, in SEG Technical Program Expanded Abstracts 2010, Society of Exploration Geophysicists, 2010, pp. 930–934.

[145] R. E. Plessix and C. Perkins, *Thematic Set: Full waveform inversion of a deep water ocean bottom seismometer dataset*, First Break, 28 (2010), pp. 71–78.

[146] G. Popov and G. Vodev, *Resonances Near the Real Axis for Transparent Obstacles*, Communications in mathematical physics, 207 (1999), pp. 411–438.

[147] R. Pratt, Z.-M. Song, P. Williamson, and M. Warner, *Two-dimensional velocity models from wide-angle seismic data by wavefield inversion*, Geophysical Journal International, 124 (1996), pp. 323–340.

[148] R. G. Pratt, *Inverse theory applied to multi-source cross-hole tomography. part 2: Elastic wave-equation method*, Geophysical Prospecting, 38 (1990), pp. 311–329.

[149] ——, *Seismic waveform inversion in the frequency domain, part 1: Theory and verification in a physical scale model*, Geophysics, 64 (1999), pp. 888–901.

[150] R. G. Pratt, C. Shin, and G. Hick, *Gauss–newton and full newton methods in frequency–space seismic waveform inversion*, Geophysical journal international, 133 (1998), pp. 341–362.

[151] R. G. Pratt and M. Worthington, *Inverse theory applied to multi-source cross-hole tomography. part 1: Acoustic wave-equation method 1*, Geophysical Prospecting, 38 (1990), pp. 287–310.

[152] Z. Qureshi, T. Ng, and G. Goodwin, *Optimum experimental design for identification of distributed parameter systems*, International Journal of Control, 31 (1980), pp. 21–29.

[153] A. U. Raghunathan and L. T. Biegler, *Mathematical programs with equilibrium constraints (mpecs) in process engineering*, Computers & chemical engineering, 27 (2003), pp. 1381–1392.

[154] J. V. Ralston, *Trapped rays in spherically symmetric media and poles of the scattering matrix*, Communications on Pure and Applied Mathematics, 24 (1971), pp. 571–582.

[155] Y. Rao and Y. Wang, *Seismic waveform tomography with shot-encoding using a restarted l-bfgs algorithm*, Scientific Reports, 7 (2017), pp. 1–9.

[156] S. Ristov, R. Prodan, M. Gusev, and K. Skala, *Superlinear speedup in HPC systems: Why and when?*, in 2016 Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, 2016, pp. 889–898.

[157] J. A. Scales, M. L. Smith, and S. Treitel, *Introductory geophysical inverse theory*, Samizdat Press Golden, 1994.

[158] S. H. Schot, *Eighty years of Sommerfeld's radiation condition*, Historia mathematica, 19 (1992), pp. 385–401.

[159] R. Seidl, *Full Waveform Inversion for Ultrasonic Nondestructive Testing*, PhD thesis, Technische Universität München, 2018.

[160] A. S. Serdyukov and A. A. Duchkov, *Hybrid kinematic-dynamic approach to seismic wave-equation modeling, imaging, and tomography*, Mathematical Problems in Engineering, 2015 (2015).

[161] R. E. Sheriff and L. P. Geldart, *Exploration seismology*, Cambridge University Press, 1995.

[162] F. Sherry, M. Benning, J. C. De los Reyes, M. J. Graves, G. Maierhofer, G. Williams, C. B. Schönlieb, and M. J. Ehrhardt, *Learning the sampling pattern for MRI*, IEEE Transactions on Medical Imaging, 39 (2020), pp. 4310–4321.

[163] A. Sinha, P. Malo, and K. Deb, *A review on bilevel optimization: from classical to evolutionary approaches and applications*, IEEE Transactions on Evolutionary Computation, 22 (2017), pp. 276–295.

[164] A. Sinha, P. Malo, P. Xu, and K. Deb, *A bilevel optimization approach to automated parameter tuning*, in Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, 2014, pp. 847–854.

[165] L. Sirgue, B. Denel, and F. Gao, *Integrating 3D full waveform inversion into depth imaging projects*, in SEG Technical Program Expanded Abstracts 2011, Society of Exploration Geophysicists, 2011, pp. 2354–2358.

[166] L. Sirgue and R. G. Pratt, *Efficient waveform inversion and imaging: A strategy for selecting temporal frequencies*, Geophysics, 69 (2004), pp. 231–248.

[167] M. A. Slawinski, *Waves and rays in elastic continua*, World Scientific, 2010.

[168] R. Snieder and J. Trampert, *Linear and nonlinear inverse problems*, in Geomatic method for the analysis of data in the earth sciences, Springer, 2000, pp. 93–164.

[169] S. Spanache, T. Escobet, and L. Travé-Massuyès, *Sensor placement optimisation using genetic algorithms*, in 15th international Workshop on Principles of Diagnosis (DX'04), Citeseer, 2004, pp. 179–184.

[170] A. Spence and I. G. Graham, *Numerical methods for bifurcation problems*, in The Graduate Student's Guide to Numerical Analysis' 98, Springer, 1999, pp. 177–216.

[171] H. v. Stackelberg et al., *Marktform und gleichgewicht*, (1934). English Translation: Theory of the market economy, Oxford University Press (1952).

[172] P. Stummer, H. Maurer, and A. G. Green, *Experimental design: Electrical resistivity data sets that provide optimum subsurface information*, Geophysics, 69 (2004), pp. 120–139.

[173] H. Sun and L. Demanet, *Extrapolated full-waveform inversion with deep learning*, Geophysics, 85 (2020), pp. R275–R288.

[174] J. Sun, S. Fomel, and L. Ying, *Low-rank one-step wave extrapolation for reverse time migration*, Geophysics, 81 (2016), pp. S39–S54.

[175] A. Tarantola, *Inversion of seismic reflection data in the acoustic approximation*, Geophysics, 49 (1984), pp. 1259–1266.

[176] A. Tarantola, *Inversion of travel times and seismic waveforms*, Seismic tomography, (1987), pp. 135–157.

[177] A. Tarantola, *Inverse problem theory and methods for model parameter estimation*, vol. 89, siam, 2005.

[178] A. Tarantola and B. Valette, *Generalized nonlinear inverse problems solved using the least squares criterion*, Reviews of Geophysics, 20 (1982), pp. 219–232.

[179] G. Thomas, *Convexity*, 2020.

[180] T. van Leeuwan, *Simple FWI*. https://github.com/TristanvanLeeuwen/SimpleFWI, 2014.

[181] L. N. Vicente and P. H. Calamai, *Bilevel and multilevel programming: A bibliography review*, Journal of Global optimization, 5 (1994), pp. 291–306.

[182] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, *Image quality assessment: from error visibility to structural similarity*, IEEE transactions on image processing, 13 (2004), pp. 600–612.

[183] A. J. Wathen, *Preconditioning*, Acta Numerica, 24 (2015), pp. 329–376.

[184] F. M. Watson, *Better imaging for landmine detection: an exploration of 3D full-wave inversion for ground-penetrating radar*, (2016).

[185] A. Wirgin, *The inverse crime*, arXiv preprint math-ph/0401050, (2004).

[186] Y. S. Wong and G. Li, *Exact finite difference schemes for solving Helmholtz equation at any wavenumber*, International Journal of Numerical Analysis and Modeling, Series B, 2 (2011), pp. 91–108.

[187] S. Xu, H. Chauris, G. Lambaré, and M. Noble, *Common-angle migration: A strategy for imaging complex media*, Geophysics, 66 (2001), pp. 1877–1894.

[188] W. Xu, M. Yuan, W. Xuan, X. Ji, and Y. Chen, *Quantitative inspection of complex-shaped parts based on ice-coupled ultrasonic full waveform inversion technology*, Applied Sciences, 11 (2021), p. 4433.

[189] Z. Xue, H. Zhu, and S. Fomel, *Full-waveform inversion using seislet regularization*, Geophysics, 82 (2017), pp. A43–A49.

[190] J. Yang, H. Zhu, X. Li, L. Ren, and S. Zhang, *Estimating P wave velocity and attenuation structures using full waveform inversion based on a time domain complex-valued viscoacoustic wave equation: The method*, Journal of Geophysical Research: Solid Earth, 125 (2020), p. e2019JB019129.

[191] H. Yu, Y. Chen, S. M. Hanafy, and J. Huang, *Visco-acoustic wave-equation traveltime inversion and its sensitivity to attenuation errors*, Journal of Applied Geophysics, 151 (2018), pp. 103–112.

[192] Z.-D. Zhang and T. Alkhalifah, *Regularized elastic full-waveform inversion using deep learning*, Geophysics, 84 (2019), pp. R741–R751.