Spring 4-15-2017

Deterministic and Probabilistic Methods for Seismic Source Inversion

Juan Pablo Madrigal Cianci
Univeristy of New Mexico

Juan Pablo Madrigal Cianci

Candidate

Mathematics and Statistics

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

Daniel Appelo , Chairperson

Mohammad Motamed

Gabriel Huerta

Deterministic and Probabilistic Methods for Seismic Source Inversion

by

Juan Pablo Madrigal Cianci

B.S Mathematics
B.S Statistics

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

Mathematics

The University of New Mexico
Albuquerque, New Mexico

May, 2017

Deterministic and Probabilistic Methods for Seismic Source Inversion

by

Juan Pablo Madrigal Cianci

B.S Mathematics, University of New Mexico, 2015

B.S Statistics, University of New Mexico, 2015

M.S Mathematics, University of New Mexico, 2017

The national Earthquake Information Center (NEIC) reports an occurrence of about 13,000 earthquakes every year, spanning different values on the Richter scale from very mild (2) to "giant earthquakes" (8 and above). Being able to study these earthquakes provides useful information for a wide range of applications in geophysics. In the present work we study the characteristics of an earthquake by performing seismic source inversion; a mathematical problem that, given some recorded data, produces a set of parameters that when used as input in a mathematical model for the earthquake generates synthetic data that closely resembles the measured data. There are two approaches to performing this source inversion: a deterministic and a probabilistic approach. We present an overview of both methods and implement them in order to perform different seismic source inversion experiments for recorded waveforms in one and two dimensions.

Acknowledgments

I would like to extend my eternal gratitude to both my advisors, Prof. Daniel Appelö and Prof. Mohammad Motamed. This project would not have been possible without their unmeasurable help.

Dedicatory

Dedicated to my mother. Everything I am and everything I have, is because of her never-ending support.

Contents

Chapter 1

Introduction

Seismic source inversion involves the set of methods which seismologists use in order to infer properties of an earthquake (such as the epicenter), through physical measurements. In practice, an array of seismographs is used to record physical information about the earthquake. Usually, the recorded data consists of time series of ground displacements, velocities, and accelerations, recorded at the surface of the ground and in observation wells. A depiction of a recording array is given in Figure (1.1).

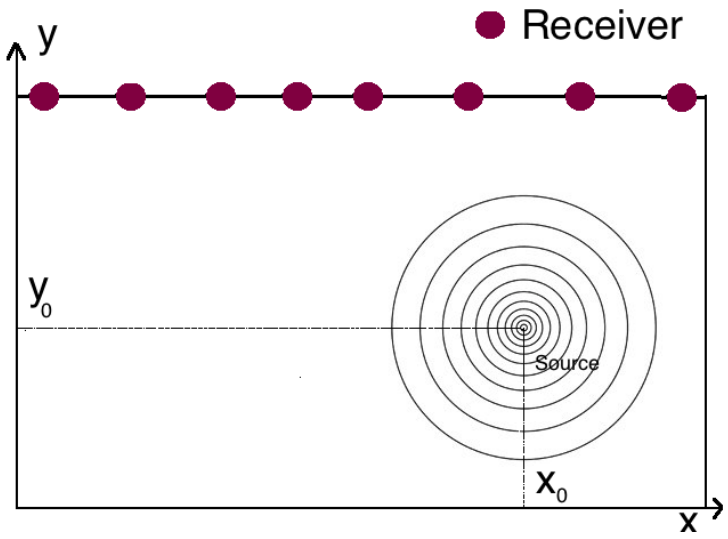

FIGURE 1.1: A schematic of the receiver array.

Once the data is collected, the goal is to obtain a set of parameters θ such that, when used as input in a forward model (e.g, a solver for the wave equation), the resulting synthetic output closely resembles the measured data. To find θ we use a cost functional $c(\theta)$ that measures the distance between the simulated and recorded data. When the cost functional is minimized we hope that θ is a good representation of the physical model.

1.1 Deterministic Vs. Probabilistic Inversion

In general, there are two approaches to find the parameters θ; deterministic and probabilistic (Bayesian), each of which has advantages and disadvantages. In the deterministic approach we use optimization algorithms, such as steepest descent, conjugate gradient, or the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method, in order to find the set of parameters θ that will minimize the cost functional. This approach is susceptible to the numerical disadvantages of the traditional optimization algorithms: 1) their convergence is sensitive to the initial guess of parameters θ_0; and 2) there is no guarantee that the method will converge to a global minimum. That being said, this deterministic approach does have the advantage of being less computationally expensive than its probabilistic counterpart.

In contrast, the probabilistic approach does not depend on an initial point. It also has the advantage of taking the uncertainty in the model into consideration. However, the main drawback of this approach is the computational cost associated with the large number of forward solves required.

1.2 Forward Simulation by Discontinuous Galerkin Methods

Regardless of which approach we take in order to solve the seismic source inversion problem, we need an accurate, stable, and reliable forward model. To this end, all forward model computations presented in this work are performed using the Discontinuous Galerkin (dG) method presented in [1].

Discontinuous Galerkin methods are element-based PDE solvers first introduced in the late seventies in order to simulate a neutron transport. As we shall discuss in Chapter 3, the main idea behind them is to divide the computational domain into k non-overlapping elements and approximate the solution to the PDE by a polynomial of degree $N_p - 1$ on each of them.

The rest of this thesis is as follows. Chapter 2 formulates the problem and discuses some of the relevant literature. Chapter 3 describes the discontinuous Galerkin method used this text. Chapter 4 and 5 describe the theory behind the deterministic and probabilistic methods, respectively. Chapter 6 presents the implementation of the methods discussed in this work. Lastly, chapter 7 presents the conclusions and discusses some future work.

Chapter 2

Seismic Source Inversion Problem

We begin our discussion with a general formulation of the seismic source inversion problem. We then discuss some of the techniques used to solve this problem, their strengths, and issues that arise with their use. We conclude this chapter with a brief discussion of related work in the literature.

2.1 Problem Formulation

The main goal of seismic source inversion is to determine a set of source parameters such that when used as input in a forward model, a good approximation to the measured data is obtained. Thus, we seek parameters θ that minimize a cost functional $c(\theta)$ measuring the misfit between the simulated and recorded waveforms at N_r recording stations,

$$c(\theta) = \frac{1}{2} \sum_{r=1}^{N_r} \int_0^T |\mathbf{d}_r(t) - \mathbf{u}(\mathbf{x}_r; \theta, t)|^2 dt. \tag{2.1}$$

Here N_r is the total number of recording stations, T is the final time, $\mathbf{x}_r$ is the location of the r^{th} receiver, $\mathbf{d}_r(t)$ is the recorded quantity, and $\mathbf{u}(\mathbf{x}_r, t)$ is the simulated quantity. Note that this cost functional is the sum of the L^2-norms of the error between the simulated and recorded waveforms at each recorder.

In the present work we consider the acoustic wave equation[1], given by

$$u_{tt} - c^2(\mathbf{x}, t)\nabla u = f(\mathbf{x}, t), \quad \mathbf{x} \in \Omega,\ t \in [0, T], \tag{2.2}$$

with initial data

$$u(\mathbf{x_0}, 0) = u_0(\mathbf{x}), \quad u_t(\mathbf{x_0}, 0) = 0, \tag{2.3}$$

and boundary conditions given by

$$\nabla u \cdot \mathbf{n} = 0, \tag{2.4}$$

where $\Omega \subset \mathbb{R}^n$ is a compact spatial domain, and the source term is given by

$$f(\mathbf{x}, t) = \mathbf{M}^T \nabla \delta(\mathbf{x} - \mathbf{x}_0) s(t; t_0, \omega, H), \tag{2.5}$$

where $\mathbf{M}$ is a vector that mimics the stress tensor coefficients of the elastic wave equation, $s(t; t_0, w, H)$ is the time component of the source term, which we take as $s(t; t_0, w) = H^2 e^{-w^2(t-t_0)^2}$; w is the frequency, H is the amplitude, t_0 is the time at which $s(t)$ obtains its peak, and $\mathbf{x_0}$ is source location. The source parameter vector θ reads

$$\theta = (\mathbf{x_0}, t_0, w, H, \mathbf{M}). \tag{2.6}$$

In practice each of the N_r receivers records data at $N_t + 1$ discrete times over $[0, T]$. That is, each seismograph will record data for each t_i, $0 = t_0 < t_1 < \cdots <$

[1] There are other more general versions of the wave equation, such as the elastic wave equation.

$t_m < \cdots < t_{N_t-1} < t_{N_t} = T$. Thus, we rewrite the integral in Eq. (2.1) as a sum

$$c(\theta) = \sum_{r=1}^{N_r} \sum_{m=0}^{N_t} |\mathbf{d}_r(t_m) - \mathbf{u}_h(\mathbf{x}_r, t_m; \theta)|^2, \tag{2.7}$$

where $\mathbf{u}_h \approx \mathbf{u}$ is a computational approximation to $\mathbf{u}$ at the grid size h. Note that we have omitted the time step Δt in Eq. (2.7) since all time steps are equal. Our goal is to find θ that minimizes $c(\theta)$.

2.2 Solution Techniques

There are in general two approaches in order to minimize Eq. (2.7), a deterministic and a probabilistic approach. The former employs optimization techniques, such as conjugate gradient or BFGS to compute a minimizer. In order to obtain a unique minimizer these types of methods add a regularization term to the cost functional. The latter approach is based on Bayes theorem and involves maximizing a likelihood function, which in turn is equivalent to minimizing the Eq. (2.7) (see Chapter 5). In this approach, the output is a probability density function for each parameter.

2.2.1 Deterministic Approaches

Deterministic approaches have the advantage of being less computationally expensive than their probabilistic counterpart. Nevertheless, according to [2], using a non-linear optimization approach for seismic source inversion remains challenging, for example:

1. Optimization algorithms might only find local minima.

2. The optimization problem might be ill-conditioned.

Adding a regularization term, such as Tykhonov or total variation regularization, can be used to address this issue (see [2]). In inverse problems, regularization refers to the process of introducing additional information in order to solve an ill-posed problem or to prevent over-fitting. We consider the following regularized cost functional:

$$J(\theta;\alpha) = \sum_{r=1}^{N_r} \int_0^T |\mathbf{d}_r(t) - \mathbf{u}_h(\mathbf{x}_r, t; \theta)|^2 dt + \alpha R(\theta), \tag{2.8}$$

where the constant α denotes the *importance* we choose for the regularization. The effect of the regularization depends on the exact form of R, for example, when R corresponds to Tykhonov regularization the cost functional becomes more quadratic. We can see an example of how regularization works in Figure (2.1).

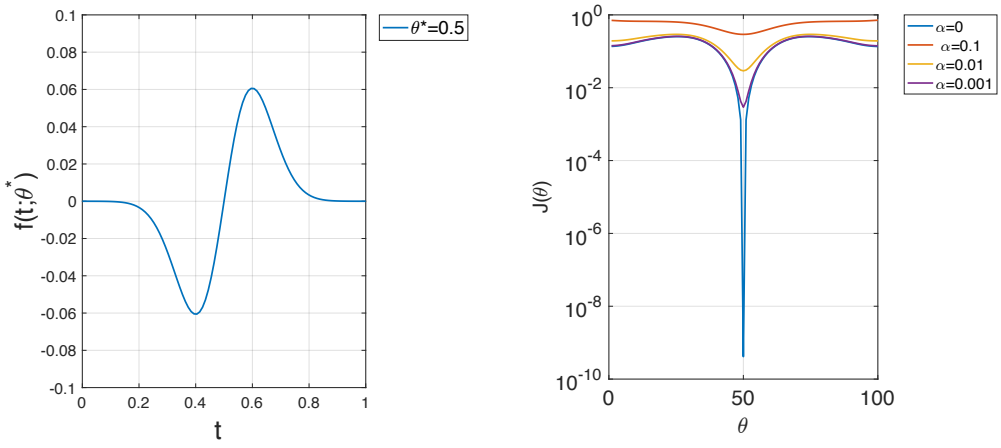

FIGURE 2.1: An illustration of how regularization works. To the left we have a toy example of a function given by Eq. (2.9). We take the true parameter to be $\theta^* = 0.5$. The right plot is the misfit function with regularization terms, for different values of α. As we can see, as α increases, the function becomes more quadratic with a single global minima.

In Figure (2.1) we illustrate the concept of regularization by considering the function

$$f(t;\theta) = (t-\theta)e^{-50(t-\theta)^2}. \tag{2.9}$$

In this toy example, we choose the true value of our parameter to be $\theta^* = 0.5$, and define the true data to be $d = f(t; 0.5)$. Having this, we can define our regularized misfit function by

$$J(\theta) = \frac{1}{2}||d - f(t;\theta)||^2 + \alpha R(\theta), \tag{2.10}$$

$$R(\theta) = \frac{1}{2}||\theta||^2. \tag{2.11}$$

In the case where $\alpha = 0$, we have that the misfit function is very steep at θ^* but becomes flatter away from this value, hence making it difficult to optimize if we do not use an appropriate initial guess. Note that as α increases, the function becomes smoother, without changing the location of the minimum. As we shall discuss later, this regularization term corresponds to imposing certain prior distributions on model parameters in the Bayesian (probabilistic) approach, which is why we do not discuss regularization in that case.

Two popular regularization techniques are Tykhonov regularization,

$$R(\theta)_{TH} = \int_{\Omega} ||\nabla\theta||^2 dx, \tag{2.12}$$

and total variation regularization,

$$R(\theta)_{TV} = \int_{\Omega} \sqrt{||\nabla\theta||^2 + \epsilon} dx, \tag{2.13}$$

for some small variation ϵ.

We can classify the deterministic methods presented in this work in two categories: gradient based and quasi-Newton methods. Gradient based methods, such as steepest descent and conjugate gradient, have previously been used to solve seismic source inversion problems in [3], and [4], where the latter authors used conjugate gradient in order to perform a magnetotelluric source inversion. Additionally, [2], implemented an optimization strategy that includes the use of conjugate gradient method. On the other hand, quasi-Newton methods, in particular the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm, have previously

been implemented in [5], in order to perform a seismic source inversion. However, contrary to the work presented here, they use the Huber norm, which is a more robust error measure that interpolates between smooth treatment of small residuals, just as an l^2 norm would, and also provides a robust treatment of large residuals, just as a l^1 norm would. Additionally, [6] used a low storage version of this method (L-BFGS) in order to perform a three-dimensional source inversion using Helicopter-born Electromagnetic data, a data collecting procedure different from the one presented in this text.

Note that both gradient-based and quasi-Newton methods require the gradient of the cost functional. We can achieve this by computing a finite difference approximation for each component of the gradient, however, this can get prohibitively expensive really quickly. A more efficient approach to computing the gradient is using adjoint state methods, which require only two simulations of the forward model. These methods have been implemented by [7], [8], and [2].

2.2.2 Probabilistic Approach

A different approach to solving source inversion problems is to consider probabilistic methods, which are based on a Bayes theorem and Markov Chain Monte Carlo (MCMC). Two major MCMC methods are the Gibbs sampler and the Metropolis-Hastings (MH) algorithm. The latter method has been used in [9] to perform the inversion of seabed reflection data in order to resolve sediment structure on the Malta Plateau (Mediterranean sea). According to the authors they were able to obtain satisfactory data consistent with other available experimental results.

The authors in [10] argue that even though the MH algorithm can handle a two-dimensional source inversion problem, it is less severe to use a Gibbs sampler, provided that the conditional distribution $\pi(\theta_j|\theta_1 \dots \theta_{j-1}, \theta_{j+1}, \dots, \theta_n)$, along with the parameter cell θ_j can be computed.

MCMC methods have the disadvantage of requiring a large number of forward solves. In order to improve the efficiency of these methods, [11], proposed a generalization of the Metropolis-Hastings algorithm which allows for parallelization. The main idea behind the method is to consider a Markov chain with more than two states, hence creating various stationary distributions and sampling from them in parallel. According to the author, this approach is widely generalizable and is fairly straight forward to implement. In addition, [12], propose the use of stochastic collocation methods based on generalized polynomial chaos (gPC) in order to construct an approximation to the forward solution over the support of a prior distribution, which in turn defines a surrogate posterior that can be evaluated repeatedly at a negligible cost. Lastly, [13], (based on previous work from [2]) propose what they call *a Stochastic Newton method* for which the MCMC is accelerated by sampling from a proposal that builds a local Gaussian approximation based on local gradient and Hessian of the log posterior.

2.2.3 Forward Wave Propagation

For this project, the forward model for the numerical solution of the wave equation is performed using discontinuous Galerkin methods. These methods have been previously implemented to solve the wave equation by [14], [15], and [16].

More recently, [1], developed a strategy for the spatial discontinuous Galerkin discretization of wave equations in second order form. The forward solver used in this thesis based on the method presented in [1].

Chapter 3

Discontinuous Galerkin Methods

Discontinuous Galerkin methods use the same element approach as finite elements, and as such we get the flexibility to discretize more complex geometries. In dG methods, the domain Ω is approximated by a combination of K non-overlapping elements D^k of the form

$$\Omega \approx \Omega_h = \bigcup_{k=1}^{K} D^k. \tag{3.1}$$

On each of these elements we express the local solution to the PDE of interest as a polynomial of order $q = N_p - 1$

$$x \in D^K : \quad u_h^k(x,t) = \sum_{n=1}^{q} \hat{u}_n^h(t)\phi_n(x) \tag{3.2}$$

where ϕ_n, is an q^{th}-degree local polynomial basis (this is called modal formulation). The global solution to a given PDE is then assumed to be approximated by the direct sum of the K piecewise n^{th}-order local polynomials solutions $u_h^k(x,t)$,

$$u(x,t) \approx u_h(x,t) = \bigoplus_{k=1}^{K} u_h^k(x,t). \tag{3.3}$$

3.1 An Illustration of a dG Method for the Transport Equation

We introduce dG methods by considering the linear, scalar, transport equation with a source term $g(x,t)$

$$\frac{\partial u}{\partial t} + \frac{\partial f(u)}{\partial x} = g(x,t), \quad x \in [L,R] = \Omega, \tag{3.4}$$

$$\tag{3.5}$$

where we choose the flux to be

$$f(u) = au. \tag{3.6}$$

The initial condition is

$$u(x,0) = u_0(x), \tag{3.7}$$

and depending on the sign of a we impose boundary conditions:

$$\begin{cases} u(L,t) = g(t), & \text{if } 0 \leq a, \\ u(R,t) = g(t), & \text{if } a < 0. \end{cases} \tag{3.8}$$

We seek an approximate solution $u_h(x,t)$ on each element D^k such that

$$u_h^k(x,t) = \sum_{n=1}^{q} \hat{u}_n^k(t)\phi_n(x) \text{ for } x \in D^k, \tag{3.9}$$

where $\{\phi_i, \ldots, \phi_n\}$ span the space of polynomials of degree at most q on D^k. We define the residual

$$R_h(x,t) = \frac{\partial u_h}{\partial t} + \frac{\partial f(u_h)}{\partial x} - g_h, \tag{3.10}$$

for some discretization of g_h, and require that this residual vanishes on each element in the Galerkin sense, i.e, we require our test function to be orthogonal to the residual,

$$0 = \int_{D^k} R_h(x,t)\phi_j^k(x)dx = \int_{D^k} \left(\frac{\partial u_h^k}{\partial t} + \frac{\partial f(u_h^k)}{\partial x} - g_h \right) \phi(x)dx. \tag{3.11}$$

We perform an integration by parts in Eq. (3.11) and obtain

$$\int_{D^k} \left(\frac{\partial u_h^k}{\partial t}\phi_j(x) - f(u)\frac{\partial \phi_j(x)}{\partial x} - g_h\phi_j(x) \right) dx = -\left[f\phi\right]_{x^k}^{x^{k+1}}. \tag{3.12}$$

Note that Eq. (3.12) implies that the solution is multiply defined for different interfaces. We define the numerical flux as the average, $u^* = \frac{u^R + u^L}{2}$, where u^R and u^L are the values of the function at the right and left sides of the interface. There are other choices for the numerical flux, for example the upwind flux (see [17]). If we perform an additional integration by parts in Eq. (3.11), we recover the strong form of the scheme

$$\int_{D^k} R_h\phi_j(x,t)dx = \left[(f_h^k - f^*)\phi_j\right]_{x^k}^{x^{k+1}}. \tag{3.13}$$

We can form the matrices M^k and S^k with components given by

$$M_{ij}^k = \int_{D^k} \phi_i(x)\phi_j(x)dx, \tag{3.14}$$

$$S_{ij}^k = \int_{D^k} \phi_i(x)\frac{\partial \phi_j(x)}{\partial x}dx, \tag{3.15}$$

in order to obtain semi-discrete schemes from Eq. (3.13) and Eq. (3.12) given by

$$M^k(u_h^k)_t - (S^K)^T(f_h) - M^k(g_h) = [f^*\phi]_{x^k}^{x^{k+1}}, \tag{3.16}$$

$$M^k(u_h^k)_t + S^K(f_h) - M^k(g_h) = [(f^* - f_h)\phi]_{x^k}^{x^{k+1}}, \tag{3.17}$$

where the weak and strong form are shown in Eq. (3.16) and Eq. (3.17), respectively. The semi-discrete schemes can be marched in time respectively by and ODE solver like RK4.

3.2 Discretization for the Scalar Wave Equation

We present a discretization for the scalar wave equation based on the formulation given in [1]. Consider the scalar wave equation in second order form with constant speed, c, and homogeneous boundary conditions:

$$\frac{\partial^2 u}{\partial t^2} = c^2\nabla^2 u + f(\mathbf{x}, t), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^d, \quad t > 0, \tag{3.18}$$

$$\alpha\frac{\partial u}{\partial t} + \beta c\nabla u \cdot \mathbf{n} = 0, \tag{3.19}$$

where α and β are nonnegative functions, $\alpha^2 + \beta^2 = 1$, and $\mathbf{n}$ is the outward unit normal. Note that $\alpha = 1$, $\beta = 0$ is our formulation of Dirichlet conditions and $\alpha = 0$, $\beta = 1$ is our formulation of Neumann conditions. Following [1], we can

rewrite Eq. (3.18) as a first order system in time

$$\frac{\partial u}{\partial t} = v, \tag{3.20}$$

$$\frac{\partial v}{\partial t} = c^2 \nabla^2 u, \tag{3.21}$$

with initial conditions

$$u(\mathbf{x}, 0) = u_0(\mathbf{x}), \quad v(\mathbf{x}, 0) = v_0(\mathbf{x}). \tag{3.22}$$

We present the discretization for one-dimensional case. Assume that the computational domain has been discretized by a uniform grid $x_0, \ldots, x_j, x_{j+1}, \ldots x_K$ with spacing h. Let $x_{j+\frac{1}{2}} = (x_j + x_{j+1})/2$. Then the mapping $z = \frac{2}{h}(x - x_{j+\frac{1}{2}})$ takes element Ω_j to the reference element $\Omega_{\mathrm{R}} = [-1, 1]$ where we expand the displacement and velocity in test functions:

$$u_j^h(x,t) = \sum_{l=0}^{q} \hat{u}_{l,j}^h(t)\phi_l(z), \tag{3.23}$$

$$v_j^h(x,t) = \sum_{l=0}^{q-1} \hat{v}_{l,j}^h(t)\psi_l(z). \tag{3.24}$$

Based on the energy function of the system $E(t)$,

$$E(t) = \frac{1}{2}\int v^2 + c^2|\nabla u|^2$$

and Eq. (3.20)-(3.21) we impose the following variational problem:

$$\int_{\Omega_j} \nabla^2 \phi_u \left(\frac{\partial u^h}{\partial t} - v^h \right) = \int_{\partial\Omega_j} \nabla \phi_u \cdot \mathbf{n} \left(\frac{\partial u^h}{\partial t} - v^* \right), \tag{3.25}$$

$$\int_{\Omega_j} \phi_v \frac{\partial v^h}{\partial t} + c^2 \nabla \phi_v \cdot \nabla u^h - \phi_v f = c^2 \int_{\partial\Omega_j} \phi_v \mathbf{w}^* \cdot \mathbf{n}, \tag{3.26}$$

$$\int_{\Omega_j} \left(\frac{\partial u^h}{\partial t} - v^h \right) = 0, \tag{3.27}$$

which can be discretized on each element by

$$M^v \hat{\mathbf{v}}'(t) + S^u \hat{\mathbf{u}}(t) = \mathcal{F}^v. \tag{3.28}$$

$$M^u \hat{\mathbf{u}}'(t) + S^v \hat{\mathbf{v}}(t) = \mathcal{F}^u, \tag{3.29}$$

where $\hat{\mathbf{u}} = [\hat{u}^h_{0,j}, \hat{u}^h_{1,j}, \ldots, \hat{u}^h_{q+1,j}]^T$, $\hat{\mathbf{v}} = [\hat{v}^h_{0,j}, \hat{v}^h_{1,j}, \ldots, \hat{v}^h_{q,j}]^T$, and the mass (M^u, M^v) and stiffness (S^u, S^v) matrices are given by

$$M^v_{k,l} = \frac{h}{2} \int_{-1}^{1} \psi_k(z)\, \psi_l(z) dz, \qquad k, l = 0, \ldots, q, \tag{3.30}$$

$$M^u_{0,l} = \frac{h}{2} \int_{-1}^{1} \phi_l(z) dz, \qquad l = 0, \ldots, q+1, \tag{3.31}$$

$$M^u_{k,l} = \frac{h}{2} \frac{4}{h^2} \int_{-1}^{1} \phi''_k(z)\, \phi_l(z) dz - \left[\frac{2}{h} \phi'_k(z)\, \phi_l(z) \right]_{-1}^{1}, \quad k = 1, \ldots, q, \quad l = 0, \ldots, q+1, \tag{3.32}$$

$$S^v_{0,l} = -\frac{h}{2} \int_{-1}^{1} \psi_l(z) dz, \qquad l = 0, \ldots, q, \tag{3.33}$$

$$S^v_{k,l} = -\frac{h}{2} \frac{4}{h^2} \int_{-1}^{1} \psi''_k(z)\, \phi_l(z) dz, \qquad k = 1, \ldots, q,\ l = 0, \ldots, q+1, \tag{3.34}$$

$$S^u_{k,l} = \frac{h}{2} \frac{4}{h^2} \int_{-1}^{1} \psi'_k(z)\, \phi'_l(z) dz, \qquad k = 0, \ldots, q, \quad l = 0, \ldots, q+1, \tag{3.35}$$

where the factors $\frac{h}{2}$, $\frac{2}{h}$ appear from the integral and derivative due to the change of variables. The flux terms are given by

$$\mathcal{F}_l^v = \left[\left(\frac{\partial u}{\partial x}\right)^* \psi_l(z(x))\right]_{x_j}^{x_{j+1}}, \quad l = 0, \ldots, q, \tag{3.36}$$

$$\mathcal{F}_l^u = -\frac{2}{h}\left[v^*(\nabla_z \phi_l(z(x)) \cdot n)\right]_{x_j}^{x_{j+1}}, \quad l = 0, \ldots, q+1, \tag{3.37}$$

and $\psi_l(z) = \phi_l(z)$ are Chebyshev polynomials:

$$\psi_l(z) = \phi_l(z) = T_l(z) = \cos(lt), \quad t = \arccos(z). \tag{3.38}$$

Once we had this semi-discrete system we used a Taylor time stepping in order to perform the time integration. For a detailed analysis of the formulation see [1].

3.3 Implementation: Wave Equation with Singular Sources

Consider the wave equation with singular sources

$$u_{tt} = u_{xx} + f(t)\delta(x - x_0) + g(t)\delta'(x - x_0).$$

We begin by considering the case when $x_0 = 0$, $g(t) = 0$ and $f(t) = -4 \times 6^2(t - 1)e^{-6^2(t-1)^2}$ so that the solution is

$$u(x,t) = \begin{cases} e^{-6^2(x+(t-1))^2} & x < 0, \\ e^{-6^2(x-(t-1))^2} & x > 0. \end{cases}$$

When x_0 coincides with a node we simply add half of the contribution from the

source to each of the numerical fluxes in such a way that the source discretization is consistent with the properties of the Dirac delta, particularly

$$\int_{-\infty}^{\infty} \psi(x)\delta(x - x_0)dx = \psi(x_0), \tag{3.39}$$

for some (integrable) function $\psi(x)$. When x_0 is inside an element we also add the source as a point-wise contribution. Both cases appear to result in spectral convergence as long as the source term integrates to 0 and this error is measured after the source has been turned off for some time.

In Figure 3.1 we plot the max-error as a function of the polynomial degree for the case when x_0 coincides with an element interface and when x_0 is inside an element. The time stepping is done by Taylor series expansion and we must therefore evaluate high order derivatives of $f(t)$. We do this by first evaluating $f(t)$ on a Chebyshev grid centered around the current timestep and then evaluating the derivatives by a finite difference approximation. As it can be seen in the figure this becomes numerically ill-conditioned due to finite precision effects when the order is too high.

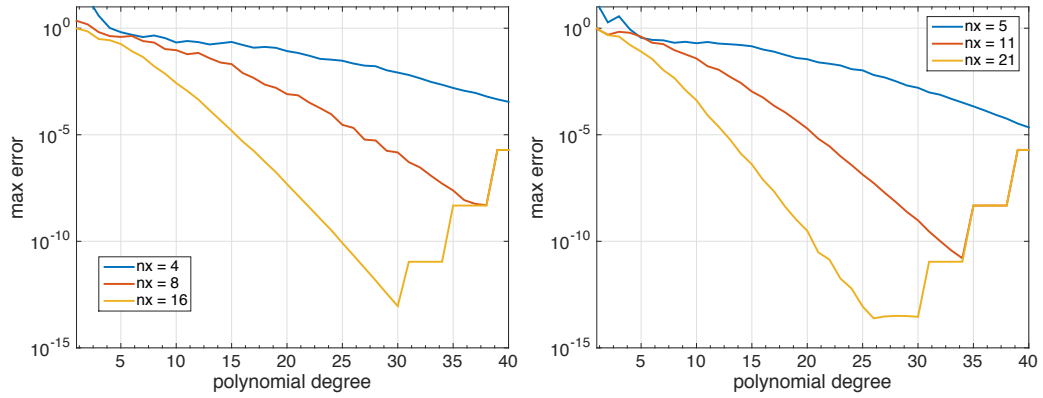

FIGURE 3.1: Max-error i.e, maximum of the error with respect to x between the computed solution $u^h(x_r, t_m)$ and the closed form solution, $u(x_r, t_m)$) as a function of polynomial degree with grid refinement. To the left is the case when the source is at an interface, to the right is when the source is inside. To the left the domain is $x \in [-5, 5]$, to the right $x \in [-5.1, 5]$. The source term for this case is $f(t)\delta(x)$, where $f(t) = -4 \times 6^2(t-1)e^{-6^2(t-1)^2}$.

Now consider the opposite case on which $f(t) = 0$ and $g(t) = -4 \times 6^2(t-1)e^{-6^2(t-1)^2}$. The solution in this case is given by

$$u(x,t) = \begin{cases} 36(x-(t-1))e^{-36(x-(t-1))^2} & x < 0, \\ 36(x+(t-1))e^{-36(x+(t-1))^2} & x > 0. \end{cases} \tag{3.40}$$

By performing an analysis similar to the first case, we obtain similar results; as we can see in Figure 3.2.

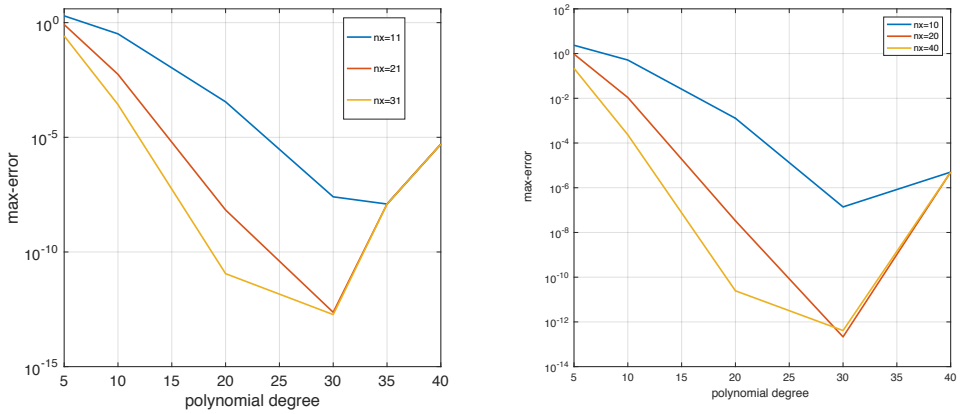

FIGURE 3.2: Max-error (i.e, maximum of the error with respect to x between the computed solution $u^h(x_r, t_m)$ and the closed form solution, $u(x_r, t_m)$) as a function of polynomial degree with grid refinement. To the left is the case when the source is at an interface, to the right is when the source is inside. To the left the domain is $x \in [-5, 5]$, to the right $x \in [-5.1, 5]$. The source term in this case is given by $g(t)\delta'(x)$, with $g(t) = -4 \times 6^2(t-1)e^{-6^2(t-1)^2}$.

Moreover, we get that the error between the analytic and numerical solution at $t = 3$ for different choices of polynomial degree q and different number of elements K is given in Figure 3.3. As we can see, as we increase q the error decreases. This should not be a surprise, for the core idea of this method is to approximate the solution u with polynomials on a finite number of elements, thus, if the polynomial degree or the number of elements increase, then the approximation u_h will be more precise.

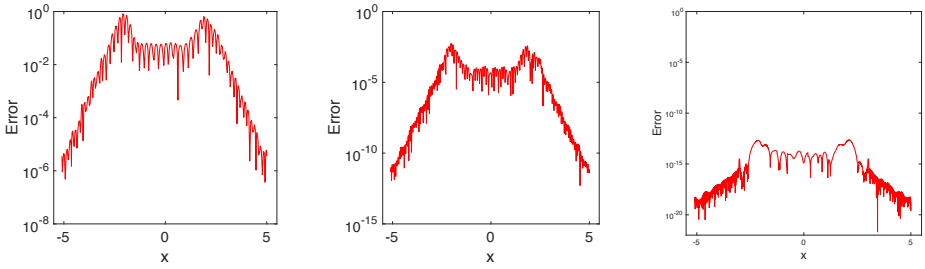

FIGURE 3.3: Error as a function of x between simulated and closed for solution at $t = 1$ for different polynomial degrees and 21 elements between [-5.1,5]. The figure at the left uses a 5-degree polynomial, whilst the middle one and the one at the right one use a 10 and 30 degree polynomial respectively. Notice the decrease with respect to the maximum error in each case.

Chapter 4

Deterministic Inversion Methods

The minimization of the cost-functional is a non-linear optimization problem, which in general is solved by an iterative algorithm. Different optimization methods make use of different information; we will consider methods that make use of the gradient as well as methods that use the gradient together with an approximation to the Hessian matrix. All of the methods rely on a line search. We begin by describing the line search algorithm we use.

4.1 The Line Search Algorithm

The backbone of the optimization methods studied in this work is the line search algorithm[1]. Assuming we know a descent direction p such that $c(\theta + \alpha p) < c(\theta)\ \forall \alpha > 0$, the line search determines a good value for α, i.e, how much we should move on that descent direction.

In general, the search direction is given by

$$p = -B\nabla c(\theta), \tag{4.1}$$

[1]There also exist other other type of algorithms that can be used instead of the line search algorithm, such as trust region. See [18]

where B is a matrix associated with different optimization methods.

We start with an adequately chosen α and shrink it iteratively by a factor of $\tau \in (0,1)$ until moving from θ to $\theta + \alpha p$ achieves an adequate decrease on the function, i.e until

$$c(\theta + \alpha p) \leq c(\theta) - \alpha m p^T \nabla c(\theta), \tag{4.2}$$

where $m \in (0,1)$. This adequate decrease condition is called the Armijo rule. The algorithm for this method is given by

Backtracking Algorithm

1. Choose α, $m \in (0,1)$, $\tau \in (0,1)$
2. While $c(\theta) - c(\theta + \alpha p) \leq -\alpha m p^T \nabla c(\theta)$, let
$$\alpha \leftarrow \alpha\tau.$$
3. Go to 2.

It is customary to choose $\tau = 1/2$ and $m << 1$ ([18]). For a detailed explanation of the convergence properties of these type of methods see [18], and [19].

4.2 Gradient-Based Algorithms

Having explained the line search algorithm, we consider different methods that arise from different choices of B_i.

4.2.1 Steepest Descent

We first consider the steepest descent algorithms. These types of methods rely on the computation of the gradient, thus making them easier to implement than other methods, however, they have the disadvantage of being inefficient. Their convergence is linear, with a constant of proportionality usually close to 1. In fact, it could happen that the method converges so slowly that $\theta_{i+1} - \theta_i$ is smaller than machine precision, which will cause the method to fail ([19]). Regardless of this issue we decided to consider them given that they provide a basis for comparison.

In order to compute the direction of steepest descent we consider the first two terms of the Taylor series expansion on $c(\theta_i + p)$

$$c(\theta_i + p) \approx c(\theta) + p^T \nabla c(\theta_i). \tag{4.3}$$

We want to choose p such that $p^T \nabla c(\theta)$ is as small as possible. Assume that $||p^T|| = 1$ and recall that

$$p^T \nabla c(\theta_i) = ||p^T|| ||\nabla c(\theta_i)|| \cos\phi, \tag{4.4}$$

which means that $p^T \nabla c(\theta)$ is minimized when p^T and $\nabla c(\theta)$ are anti-parallel. This implies that

$$p^T = \frac{-\nabla c(\theta_i)}{||\nabla c(\theta_i)||}. \tag{4.5}$$

The steepest descent method can be implemented form this result by iteratively doing a line search in the direction of p. The steepest descent algorithm is given in in the following table:

Steepest Descent Algorithm

1. Choose an initial point θ_0

   and some convergence tolerance, ϵ.

2. For $i = 0, 1, \ldots,$

   (a) if $||\nabla c(\theta_i)|| < \epsilon$, stop. Otherwise,

   (b) Set $p_i = -\nabla c(\theta)$

   (c) use a line search to determine $\theta_{i+1} = \theta_i + \alpha_i p_i$.

4.2.2 Conjugated Gradients

We now present an improved version of steepest descent. We first discuss conjugate gradients for linear optimization and then move to the non-linear case. Consider the quadratic linear function

$$f(\theta) = \frac{1}{2}\theta^T A\theta - b^T\theta. \tag{4.6}$$

We want to generate a sequence of vectors $\{p_i\}$ that are conjugate with respect to the coefficient matrix A, i.e, we want

$$p_i^T A p_j = 0 \text{ if } j \neq i. \tag{4.7}$$

Let θ be a linear conbination of N of these vectors;

$$\theta = \sum_{j=0}^{N} \alpha_j p_j. \tag{4.8}$$

Thus, we can write Eq. (4.6) as

$$f(\theta) = f\left(\sum_{j=0}^{N} \alpha_j p_j\right) \tag{4.9}$$

$$= \frac{1}{2}\left(\sum_{i=0}^{N} \alpha_j p_i\right)^T A\left(\sum_{j=0}^{N} \alpha_j p_j\right) - b^T\left(\sum_{j=0}^{N} \alpha_j p_j\right) \tag{4.10}$$

$$= \frac{1}{2}\sum_{i=0}^{N}\sum_{j=0}^{N} \alpha_i \alpha_j p_i^T A p_j - \sum_{i=0}^{N} \alpha_i b^T p_i \tag{4.11}$$

$$= \frac{1}{2}\sum_{j=0}^{N} a_j^2 p_j^T A p_j - \sum_{i=0}^{N} \alpha_i b^T p_i = -\sum_{i=0}^{N}\left(\frac{1}{2} a_j^2 p_j^T A p_j - \alpha_j b^T p_j\right). \tag{4.12}$$

Thus, if we want to minimize $f(\theta)$ over all α, we can take the derivative of Eq. (4.12) with respect to α and set it equal to 0,

$$a_j p_j^T A p_j - b^T p_j = 0 \implies \alpha_j = \frac{b^T p_j}{p_j^T A p_j}. \tag{4.13}$$

If we let the residual at the i^{th} iteration to be $r_i = b - Ax_i = -\nabla f(\theta)$ and replace the computation of α with a line-search algorithm, we are able to use this method to minimize non-linear function (see [18]). Thus, the conjugate gradient algorithm is given by:

Conjugate Gradient Algorithm

1. set $p_{-1} = 0, \beta_0 = 0$, choose some initial points θ_0, and some convergence tolerance, ϵ.
2. for $i = 0, 1, \ldots,$
   (a) if $||\nabla c(\theta_i)|| < \epsilon$, stop. Otherwise,
   (b) if $i > 0$, let
   $$\beta_i = \frac{\nabla c(\theta_i)^T \nabla c(\theta_i)}{\nabla c(\theta_{i-1})^T \nabla c(\theta_{i-1})}.$$
   (c) Set $p_i = -\nabla c(\theta_i) + \beta_i p_{i-1}$.
   (d) use a line search to determine $\theta_{i+1} = \theta_i + \alpha_i p_i$.

In general there are different formulas for β_i. The one in the algorithm is called the Fletcher-Reeves, however, other formulas for β_i include

$$\beta_i^{FR} = \frac{\nabla c(\theta_i)^T \nabla c(\theta_i)}{\nabla c(\theta_{i-1})^T \nabla c(\theta_{i-1})}, \tag{4.14}$$

$$\beta_i^{PR} = \frac{\nabla c(\theta_i)^T (\nabla c(\theta_i) - \nabla c(\theta_{i-1})}{\nabla c(\theta_{i-1})^T \nabla c(\theta_{i-1})}, \tag{4.15}$$

$$\beta_i^{HS} = -\frac{\nabla c(\theta_i)^T (\nabla c(\theta_i) - \nabla c(\theta_{i-1})}{s_{n-1}^T \nabla c(\theta_{i-1})^T \nabla c(\theta_{i-1})}, \tag{4.16}$$

$$\beta_i^{DY} = \frac{\nabla c(\theta_i)^T \nabla c(\theta_i)}{s_{n-1}^T \nabla c(\theta_{i-1})^T \nabla c(\theta_{i-1})}, \quad , \tag{4.17}$$

where the superscripts stand for FR: Fletcher-Reeves, PR: Polak-Ribiere, HS Hestens-Stieffel, and DY: Dai-Yuan. Additionally,

$$s_n^T = \nabla c(\theta_i) + \beta_i s_{n-1}, \tag{4.18}$$

$$s_0^T = \nabla c(\theta_0). \tag{4.19}$$

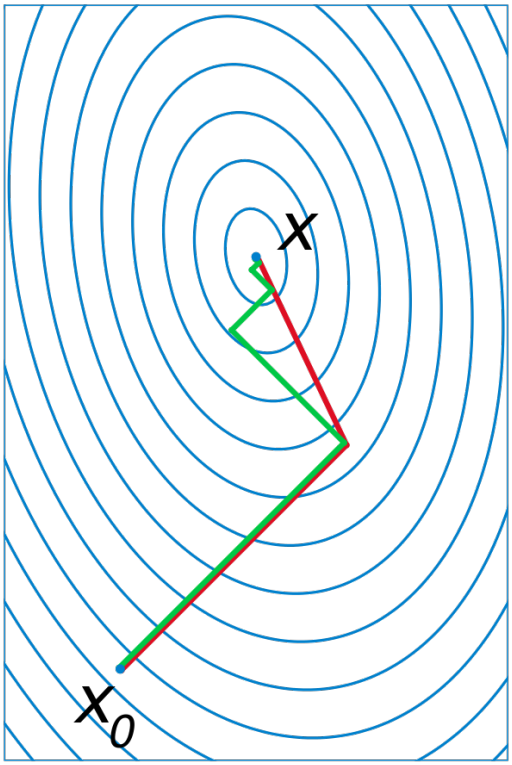

FIGURE 4.1: Comparison between the conjugate gradient and the steepest descent algorithms, where steepest descent is plotted in green and the conjugate gradient is plotted in red.

Finally, we illustrate a comparison between both methods in Figure 4.1, where steepest descent is plotted in green and the conjugate gradient is plotted in red. As we can see, the conjugate gradient is a more efficient way to obtain the minimum; note that the steepest descent has a zig-zagging trajectory, whilst the conjugate gradient is more direct.

4.3 Quasi-Newton Methods

We now discuss quasi-Newton methods. The main idea behind these methods is to perform a Taylor expansion of the function c in the vicinity of θ, i.e

$$c(\theta + p) \approx c(\theta) + p^T \nabla(c(\theta)) + \frac{1}{2} p^T H(\theta) p, \tag{4.20}$$

where $H(\theta)$ is the Hessian matrix of the function $c(\theta)$. In general, a necessary condition for a local minimum of our approximation is given when

$$\nabla c(\theta) + H(\theta)p = 0, \implies p = -(H(\theta))^{-1} \nabla c(\theta), \tag{4.21}$$

which is called the *Newton direction*. In general, an exact Newton method for minimization is reliable when the Hessian matrix exist and is positive definite. However, as we have mentioned before, this matrix is not necessarily easy to compute, and as such, we resort to approximate this matrix and and proceed to modify this approximation iteratively. In general we can think of these types of methods as a generalization of the secant method for one dimensional problems.

4.3.1 The BFGS Algorithm

Of peak importance is the BFGS algorithm, developed independently in 1970 by Broyden, Fletcher, Goldfarb, and Shanno (see [20]), this method provides a rank-two update formula for the approximation of the Hessian. Denote the approximation of the Hessian at iteration k by B_k. From Eq. (4.21), we get that our search direction at the k-th iteration satisfies the Newton equation

$$B_k p_k = -\nabla c(\theta_k) \implies p_k = -B_k^{-1} \nabla c(\theta_k). \tag{4.22}$$

Once we have obtained our search direction, we proceed to do a line search in the direction of p_k in oder to obtain the next point in our iteration, θ_{k+1}. In general, we want to update B_{k+1} by adding two rank-one matrices, that is

$$B_{k+1} = B_k + U_k + V_k. \tag{4.23}$$

Moreover, given its computational properties, as well as its similarity to the Hessian, it is desirable for B_k to be a symmetric positive definite matrix. One way to assure this is to consider $U_k = \alpha u u^T$ and $V_k = \beta v v^T$. Thus, the update for B_k at the $k+1$-step is given by

$$B_{k+1} = B_k + \alpha u u^T + \beta v v^T. \tag{4.24}$$

Define the quasi-Newton condition at the k-th step by

$$B_{k+1}(\theta_{k+1} - \theta_k) = \nabla c(\theta_{k+1}) - \nabla c(\theta_k), \tag{4.25}$$

$$\implies B_{k+1} d_k = y_k, \tag{4.26}$$

where $s_k = \theta_{k+1} - \theta_k, y_k = \nabla c(\theta_{k+1}) - \nabla c(\theta_k)$. Thus, if we let

$$u = y_k \tag{4.27}$$

$$v = B_k s_k, \tag{4.28}$$

we get that,

$$\alpha = \frac{1}{y_k^T s_k}, \quad \beta = \frac{1}{s_k^T \beta_k s_k}, \tag{4.29}$$

which in turn implies that

$$B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T \beta_k s_k}. \tag{4.30}$$

There are different ways to compute the first approximation to the Hessian, B_0, however the identity matrix I is usually a good enough approximation. The general form for the BFGS algorithm is given by

BFGS Algorithm

1. Choose an initial guess θ_0 and and initial guess for the Hessian, B_0. If no information is available, I is usually a good choice.
2. for $k = 0, 1, \ldots,$
   (a) If $||\nabla c(\theta_i)|| < \epsilon$, stop. Otherwise,
   (b) Solve $B_k p_k = \nabla c(\theta_k)$ for p_k.
   (c) Determine $\theta_{k+1} = \theta_k + \alpha p_k$ using a line-search
   (d) Compute $s_k = \theta_{k+1} - \theta_k$ and $y_k = \nabla c(\theta_{k+1}) - \nabla c(\theta_k)$.
   (e) Compute $B_{k+1} = B_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{B_k s_k s_k^T B_k}{s_k^T \beta_k s_k}$.

It is important to mention that this is the simplest form of this method; there are versions of the BFGS algorithm that are optimized in order to minimize the storage used, which is useful for large problems. A more detailed discussion of these methods is given in [19], and [18].

4.4 Adjoint State Methods

We finalize this chapter by discussing adjoint state methods, which are efficient ways of computing the gradient or Hessian. For our particular problem of interest, these methods can be understood as obtaining information in two ways; one when we solve for our wave-field, i.e record the data from one source at N_r locations, and the other from reversing the problem and solving for the case where we have one recorder (at the location of the source) and N_r sources (at the location of the recorders). This section draws from the papers by [2], and [8]. We will discuss both implementations.

The idea behind the adjoint computation of the gradient presented in [2], is as follows. Denote the forward solver of our model by $L_\theta u = f$ (where the subscript denotes an intrinsic dependency on θ and f denotes the forcing term) and the minimizer as $F(u, \theta)$, where u is the continuous wave-form. Moreover, define the Lagrangian function by

$$\mathcal{L}(\theta, u, p) = F(u) + p^T(Lu - f), \tag{4.31}$$

where p is a Lagrange multiplier but can be thought of as the adjoint wave-field. Minimizing the function thus gives

$$\nabla_u \mathcal{L} = \nabla F + L^T p = 0 \implies p^T = (\nabla F)^T L^{-1}, \tag{4.32}$$

$$\nabla_p \mathcal{L} = Lu - p = 0 \implies u = L^{-1} f, \tag{4.33}$$

$$\nabla_\theta \mathcal{L} = p^T \left[D_\theta L \right] u = 0, \tag{4.34}$$

where $D_x f(\Delta x) = \lim_{h\to 0} \frac{f(x+h\Delta x)-f(x)}{h}$. Thus, replacing from Eq. (4.32) and Eq. (4.33) we get that Eq. (4.34) becomes

$$\nabla_\theta \mathcal{L} = (\nabla F)^T L^{-1} \left[D_\theta L\right] L^{-1} f. \tag{4.35}$$

Note that there are two key factors here; the term $L^{-1}f$ corresponds to a forward solve and $(\nabla F)^T L^{-1}$ corresponds to a backwards solve for the adjoint state using the difference between the the measured and simulated solutions for the (forward in time) model.

On the other hand, [8] present a slightly different approach for the computation of the gradient via adjoint state methods. Just as in the previous approach, define the adjoint state waveform by p. From Theorem 1 in [8], we have that for a conservative discretization,

$$\sum_{n=1}^{N_t} \langle G, u\rangle = \sum_{n=1}^{N_t} \langle p, H\rangle \tag{4.36}$$

where H and G the source terms for u and p respectively. Eq. (4.36) can be thought of as a reversibility condition.

Recall the formula for the (discrete) cost functional

$$c(\theta) = \frac{1}{2}\sum_{r=1}^{N_r}\sum_{m=0}^{N_t} |d_r(t_m) - u_h(\mathbf{x}_r, t_m; \theta)|^2, \tag{4.37}$$

thus, assuming that we want to invert for n parameters, the gradient of the cost functional will look like

$$\nabla c(\theta) = \left(\frac{\partial k}{\partial \theta_1} \dots \frac{\partial k}{\partial \theta_n} \right) \implies \tag{4.38}$$

$$\frac{\partial c}{\partial \theta_j} = \sum_{r=1}^{N_r} \sum_{m=0}^{N_t} \left\langle |d_r(t_m) - u_h(\mathbf{x}_r, t_m; \theta)|, u_{\theta_j} \right\rangle . \tag{4.39}$$

Note that equation Eq. (4.39) can be used to compute the gradient, however, this would imply that we need to run the forward model once for each parameter, which is something that we might want to avoid, particularly if the forward model is expensive to compute. Thus, if we define $G = |d_r(t_m) - u_h(\mathbf{x}_r, t_m; \theta)|$ and use the reversibility condition Eq. (4.36), we obtain that

$$\frac{\partial c}{\partial \theta_j} \approx \sum_{r=1}^{N_r} \sum_{m=0}^{N_t} \left\langle G, u_{\theta_j} \right\rangle = \sum_{r=1}^{N_r} \sum_{m=0}^{N_t} \left\langle p, H_{\theta_j} \right\rangle . \tag{4.40}$$

Thus, we can compute each component of the gradient by solving the forward model and the adjoint forward model, and then computing an inner product which will be of negligibly cost compared to the forward model solution. Note that this does not depend much on the number of components of θ. Given the reversibility condition, we can think of the adjoint forward model as the standard forward model traveling backwards in time, using the misfit as the source function.

Chapter 5

Bayesian Inversion

In this chapter we present the theory behind Bayesian inversion, an alternative to the deterministic approach to finding θ. This method is largely based on Bayes theorem.

5.1 Bayes Formulation

5.1.1 Bayes Theorem

Denote the probability of two events, A, B by $\pi(A), \pi(B)$, respectively. Moreover, let $\pi(A|B)$ denote the probability of event A given event B (this is called conditional probability). Then, Bayes theorem states that

$$\pi(A|B) = \frac{\pi(B|A)\pi(A)}{\pi(B)}. \tag{5.1}$$

Throughout this document we will refer to $\pi(A)$ as the *prior* and $\pi(A|B)$ as the *posterior*. In practice, we are interested in finding the distribution of the posterior, which we will be able to estimate provided that we can compute the right hand side of equation Eq. (5.1).

5.1.2 Construction of the Posterior

In practice, Bayesian inversion considers the prior as expert information; that is, knowledge that we have *a priori* about the parameters based on previous results or expert information. Assuming that the noise is additive, the recorded data $\mathbf{d}_r(t_m)$ is given by

$$\mathbf{d}_r(t) = \mathbf{u}(\mathbf{x}_r, t; \theta) + \mathbf{E}, \tag{5.2}$$

where $\mathbf{E} \in \mathbb{R}^{N_r}$ is the noise of the measurements assumed to be $E_i \sim^{iid} N(0, \sigma^2)$, with $i = 1, \dots N_r$, and $\mathbf{u}(\mathbf{x}_r, t; \theta)$ is the true quantity given by the deterministic model.

By equation Eq. (5.1), our posterior distribution takes the form

$$\pi(\theta|\mathbf{d}_r(t_m)) \propto \pi(\mathbf{d}_r(t_m)|\theta)\pi(\theta). \tag{5.3}$$

In order to propose a likelihood function and a posterior distribution we follow [21], and write

$$\pi(\theta|\mathbf{d}(t_m)) \propto \exp\left[-\frac{1}{2}\sum_{r=1}^{N_r}\sum_{m=0}^{N_t}\frac{|\mathbf{d}_r(t_m) - \mathbf{u}_h(\mathbf{x}_r, t_m; \theta)|^2}{\sigma^2}\right]\pi(\theta), \tag{5.4}$$

where σ is the standard deviation of the noise. Note that the likelihood becomes a normal distribution since the noise is assumed to be normally distributed. Based

on this, we can also define the log-cost functional, $L(\theta)$:

$$\begin{aligned} L(\theta) &:= -\log(\pi(\theta|\mathbf{d}_r(t_m))) \\ &= \frac{1}{2}\sum_{r=1}^{N_r}\sum_{m=0}^{N_t}\frac{|\mathbf{d}_r(t_m) - \mathbf{u}_h(\mathbf{x}_r, t_m; \theta)|^2}{\sigma^2} - \log(\pi(\theta)) + c, \end{aligned} \tag{5.5}$$

for some constant c. It is evident then that maximizing the likelihood is an equivalent problem to minimizing our log-cost functional. Thus, we get an expression for our posterior which we will be able to use provided that we can sample from the right hand side of equation Eq. (5.5). This is where the Markov Chain Monte Carlo algorithm come in handy.

5.2 MCMC Sampling: Metropolis Hastings

Markov chain Monte Carlo (MCMC) algorithms are used for sampling from some probability distribution based on the construction of a Markov chain which has the property that its stationary distribution is the same as the distribution that we are trying to sample from. That is, given a probability distribution π, we are able to draw samples form it by constructing a Markov chain P with stationary distribution π ($P^n \to \pi$ as $n \to \infty$), and then drawing samples from it. From this requirement of stationarity our approximation becomes better as the number of steps n becomes lager.

As a motivating example consider computing the right hand side of equation Eq. (5.1). Assuming we have $\pi(B|A), \pi(A)$, and $\pi(B)$ this should be a straightforward computation. However, if we do not know the value of $\pi(B)$ we would have to

compute it by

$$\pi(B) = \int_{\Omega} \pi(B, A) dA, \tag{5.6}$$

where Ω_A is the parameter space of A. Such integral is not necessarily easy to calculate, and we are not generally able to compute it analytically, and as such we need to resort to a numerical approximation, thus making the MCMC methods helpful for this task.

There are different types of MCMC algorithms, however, the Metropolis and the Metropolis-hastings algorithms are of particular importance to the present work. The Metropolis algorithm was first presented by [22], and was then generalized to non-symmetric distributions by [23].

Intuitively, the Metropolis-Hastings algorithm generates a chain of values such that, as we increase the length of the chain, the distribution of these values approximates the targeted distribution. These sample values are produced iteratively, with the distribution of the next sample being dependent only on the current sample value, effectively making the sequence of samples a Markov chain. The algorithm picks a candidate based on the current sample value, and this candidate is later accepted or rejected with some probability a. If we reject the candidate, the current value of the chain remains unchanged, and if we accept, this generated value becomes the new chain value.

5.2.1 Theory

We follow the approach presented in [24] in order to show the theoretical framework behind the Metropolis-Hastings algorithm. Let A, $A \subset \mathcal{R}^d$ be our space of parameters. For $x, y \in A$, we define $p(x, y)$ as the transition probability function

from x to y. We say that $p(x, y)$ satisfies the detailed balance condition if

$$\pi(x)p(x, y) = \pi(y)p(y, x), \qquad \forall x, y \in A. \tag{5.7}$$

Consider the proposal distribution, $q(x, y)$, which generates a value y whenever the process is at point x. Usually $q(x, y)$ is not reversible, and as such, we can assume without of generality that

$$\pi(x)q(x, y) > \pi(y)q(y, x), \tag{5.8}$$

which, intuitively, corresponds to the case on which we are moving from x to y more frequently than from y to x. Introduce some probability $a(x, y) \leq 1$ so that

$$\pi(x)q(x, y)a(x, y) \geq \pi(y)q(y, x)a(y, x). \tag{5.9}$$

We want to choose $a(x, y)$ in such a way that the reversibility condition holds for Eq. (5.9). In order to do so, we must choose $a(y, x) = 1$ and $a(x, y)$ as small as possible:

$$\pi(x)q(x, y)a(x, y) = \pi(y)q(y, x)a(y, x), \tag{5.10}$$

$$= \pi(x)q(x, y)a(x, y) = \pi(y)q(y, x)(1), \tag{5.11}$$

$$\implies a(x, y) = \frac{\pi(y)q(y, x)}{\pi(x)q(x, y)}. \tag{5.12}$$

Lastly, since $a(x, y) \leq 1$ $a(x, y)$, let

$$a(x, y) = \begin{cases} \min\left[\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\right] & \text{if } \pi(x)q(x, y) > 0 \\ 1 \text{ otherwise,} \end{cases} \tag{5.13}$$

which will in turn guarantee the detailed balance condition. Note that in the special case where $q(x, y)$ is symmetric, i.e, $q(x, y) = q(y, x)$, Eq. (5.26) becomes

$$a(x, y) = \begin{cases} \min\left[\frac{\pi(y)}{\pi(x)}, 1\right] & \text{if } \pi(x)q(x, y) > 0 \\ 1 \text{ otherwise,} \end{cases} \tag{5.14}$$

which was the original version of the algorithm proposed by [22].

5.2.2 Numerical Algorithm

Having described the theory behind the Metropolis-hastings algorithm, we proceed to take the above section into an algorithmic form:

Metropolis Hastings Algorithm

1. Set an initial value $\theta^{(1)}$
2. For $i = 1, \ldots, N$
3. Given θ^i, generate θ^{i+1} from $q(\theta^i, \theta^{i+1})$
4. Compute

$$a(\theta^{i+1}, \theta^i) = \begin{cases} \min\left[\frac{\pi(\theta^{i+1}|\mathbf{d})q(\theta^i,\theta^{i+1})}{\pi(\theta^i|\mathbf{d})q(\theta^{i+1},\theta^i)}, 1\right] & \text{if } \pi(\theta^{i+1})q(\theta^{i+1}, \theta^i) > 0, \\ 1 \text{ otherwise (i.e, if they are outside the support).} \end{cases}$$

5. Sample $\hat{U}$ from the uniform distribution $\hat{U} \sim U(0, 1)$.
6. If $\hat{U} \leq a(\theta^{i+1}, \theta^i)$, then $\theta^{i+1} = \theta^{i+1}$. Otherwise , $\theta^{i+1} = \theta^i$.
7. End.

Intuitively, we wonder around the support of the posterior, visiting the points with higher probability density more frequently. Note the following remarks:

- It is common to run these algorithms for a large number of iterations and remove the first N_d samples. This is called the *burn-in* period. This is done in order to loose the dependency on the initial points and guarantee the detailed balance condition.
- Two good diagnostic plots are the autocorrelation function and the plots of the traces. In general we would like to see that there is no clear pattern or dependence, for this would imply that the Markov-chain was not stationary.

5.2.3 Proposal Distribution

As described above, the MH algorithm relies on simulating a candidate value of θ^{i+1} from a proposal distribution, $q(\theta^i, \theta^{i+1})$, which in turn, will be later accepted with some probability a. In practice, symmetric proposal distributions are easier to implement because this implies that $q(\theta^{i+1}, \theta^i) = q(\theta^i, \theta^{i+1})$ thus simplifying the ratio $a(\theta^{i+1}, \theta^i) = \frac{\pi(\theta^{i+1}|\mathbf{d})}{\pi(\theta^i|\mathbf{d})}$. Examples of symmetric proposals include the normal and uniform distribution. In general, this proposal distribution will dictate how our algorithm behaves, since the the acceptance rate depends on how *wide* the distribution is. As an example, consider the case where the proposal distribution is normal with variance σ_p^2. If σ_p^2 is too small, the acceptance rate will be close to 1, however, there will be point in the support that would not be explored. On the other hand, if σ_p^2 is too big, the acceptance rate would be very small and the chain would hardly move to a different point.

5.2.4 A Toy Example

We implement the Metropolis-Hastings algorithm on a simple example. Suppose we have some data that looks like the following figure

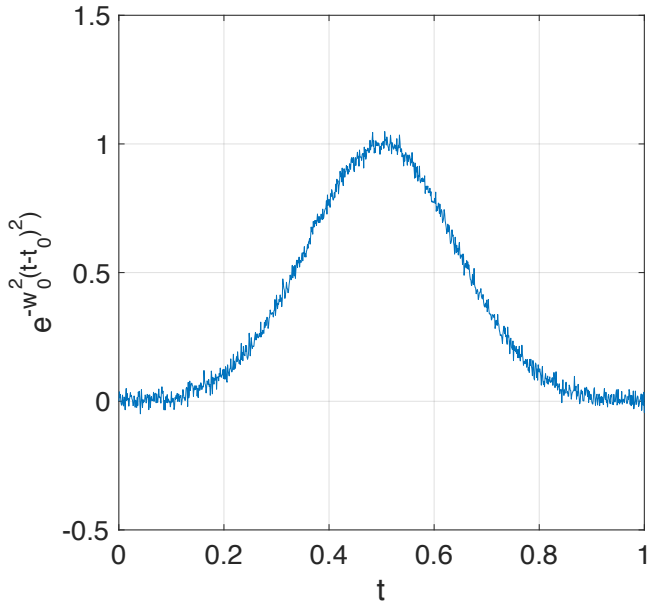

FIGURE 5.1: "Recorded" data for our toy example.

Moreover, suppose we know that the data is modeled by the function

$$f(t; t_0, w_0) = e^{-w_0(t-t_0)^2}, \tag{5.15}$$

however, we do not know what values of t_0 and w_0 were used to produce Figure 5.1. Additionally, we would like to characterize the distribution of the noise. Choosing a normal proposal distribution and the following prior distributions,

$$t_0 \sim U(0,1), \quad w_0 \sim U(0,10), \quad \sigma^2 \sim \Gamma^{-1}(2,100), \tag{5.16}$$

simulating 20000 times, and considering only the last 10000 iterations (called a burn-in period), we obtain the results shown in Figure 5.2.

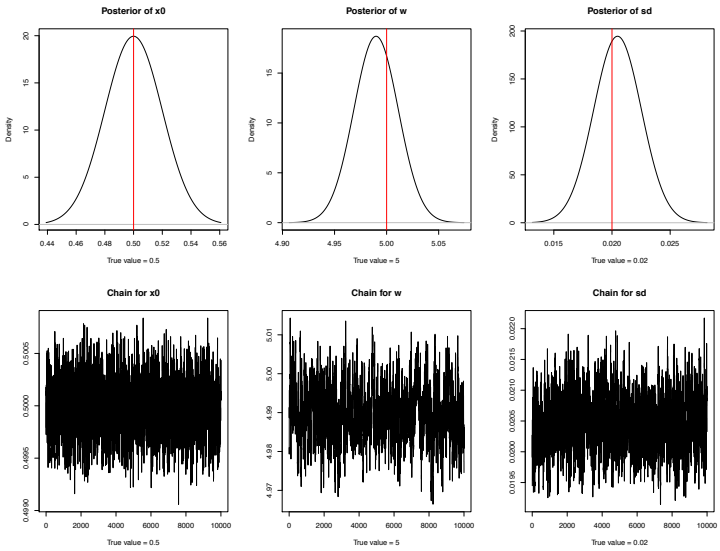

FIGURE 5.2: Posterior distribution for our parameters of interest.

As we can see, we are able to obtain smooth posterior distributions centered very close to the true value of the parameter. Moreover, there is a lot of mixing in the chain, which implies that the reversibility condition of the MCMC was met. There is no visible trend in the lower plots, which is ideal.

5.2.5 Parallelization of the Metropolis Hastings Algorithm

In order to address the last concern, we were able to come up with a rather simple, yet effective approach to increase the computational efficiency. Based on ideas proposed by [11], and [25], we proceed to parallelize the MH algorithm to some extent. The idea behind the method is rather simple; instead of running the MCMC algorithm for a large number of iterations N and collect the same amount of samples, we proceed to run the method simultaneously on p different machines (in practice processors) for a smaller number of iterations, N_p. After doing so, we

proceed to combine all p samples, obtaining N elements in total, thus effectively increasing the efficiency of the method. We should be a little careful on how we approach to do this because, in practice we discard the first N_d observations when we perform MCMC algorithms in order to eliminate any dependencies to the initial point and obtain a good mixing of the chain (i.e, obtaining an irreversibly Markov chain). Thus, we run the MCMC algorithm for N iterations of which we only consider the last $M = N - N_d$ of them. Define the CPU gain, G, by

$$G = \frac{N + N_d}{\frac{N}{p} + N_d} = \tag{5.17}$$

$$p\frac{N + N_d}{N + N_d p} \tag{5.18}$$

$$\approx p, \text{ for large } N. \tag{5.19}$$

Thus, we can obtain roughly a p-times gain in CPU time for large N. This simple procedure allows us to tackle problems with an expensive forward model on an efficient way.

Gelman-Rubin Convergence

A diagnostic tool that can be used to examine the validity of the parallelized MH is the Gelman-Rubin convergence. Proposed by [26], this diagnostic analyzes the difference between multiple Markov chain by examining the variance between and within chains. Suppose we have M chains of length L. Denote the l^{th} simulated chain by θ_l and let $\hat{\theta}_l$, $\hat{\sigma}_l^2$ be the sample posterior mean and variance of this chain. We define the sample mean over all chains by $\bar{\theta} = (1/M)\sum_{l=1}^{M}\hat{\theta}_l$. Then we can define the between-chains variance, given by

$$B = \frac{L}{M-1}\sum_{l=1}^{M}(\bar{\theta} - \hat{\sigma}_l)^2, \tag{5.20}$$

and the within-chains variance, given by

$$W = \frac{1}{M} \sum_{l=1}^{M} \sigma_l^2. \tag{5.21}$$

We define an unbiased estimator for the posterior variance of θ:

$$\hat{V} = \frac{L-1}{L} W + \frac{M+1}{ML} B, \tag{5.22}$$

and define the potential scale reduction factor, R,

$$R = \sqrt{\frac{df+3}{df+1} \frac{\hat{V}}{W}}, \tag{5.23}$$

where df are the degrees of freedom of a t-distribution. Ideally, we want R to be as close to 1 as possible. Large values of R indicate that the chain needs more iterations in order to converge (see [27], and [28]). We implement the Gelman-Rubin convergence to the toy model presented in Section 5.2.4 in order to evaluate the behavior of the parallelized chains:

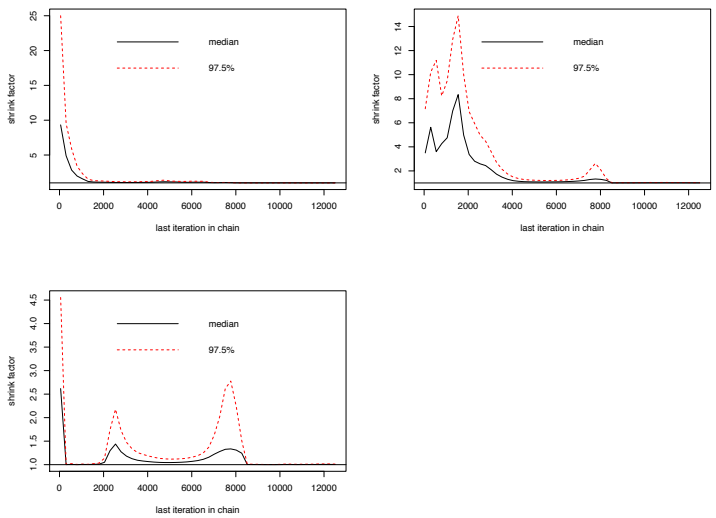

FIGURE 5.3: Gelman-Rubin convergence plot. On the top we have the convergence for t_0 and w_0, from left to right. At the bottom we have the convergence for σ^2.

As we can see from Figure 5.3, the chains approach 1, which indicates convergence between the parallelized chains.

5.3 Other Methods

We finalize this chapter by describing one last approach to perform the Bayesian inversion. These types of methods are based on generalized polynomial chaos (gPC), a technique that creates a polynomial approximation to the likelihood function and as such, to the posterior. These methods are in general more computationally efficient than MCMC algorithms provided that the simulated quantity is highly regular with respect to the parameter vector. We follow an approach similar to [12]. An efficient way to implement this gPC technique is to use stochastic collocation. One of the main advantages of this method is that it requires a finite number of uncoupled deterministic solvers, thus increasing the efficiency of the

inversion. For a detailed analysis of the convergence of these type of methods see [12].

5.3.1 Generalized Polynomial Chaos

We denote the forward model by $F(\theta)$. We define the N^{th} degree gPC expansion of $F(\theta)$ by

$$F_N(\theta) = \sum_{|\mathbf{i}|=0}^{N} a_{\mathbf{i}} \phi_{\mathbf{i}}(\theta), \tag{5.24}$$

where $\mathbf{i} = (i_1, \ldots, i_m)$ is a multi-index with $|\mathbf{i}| = i_1 + \cdots + i_m$, $\phi_{\mathbf{i}}(\theta_l)$ defined as $\phi_{\mathbf{i}}(\theta) = \prod_{l=i_1}^{i_m} \phi_l(\theta)$ are l^{th} degree orthogonal polynomials satisfying

$$\int \phi_m(\theta_k)\phi_n(\theta_k)\pi(\theta_k)d\theta_k = \delta_{m,n}, \tag{5.25}$$

and a_i is given by

$$a_{\mathbf{i}} = \int_{\Theta} F(\theta)\phi_{\mathbf{i}}(\theta)\pi(\theta)d\theta. \tag{5.26}$$

Furthermore we have that this expansion converges when $F_N(\theta)$ is square integrable with respect to the prior, i.e

$$||F(\theta) - F_N(\theta)||^2 = \int (F(\theta) - F_N(\theta))^2 \pi(\theta)d\theta \to 0, \text{ as } N \to \infty. \tag{5.27}$$

Finally, it can also be shown that the convergence rate will depend on the regularity of F; when F is smooth then the converge rate will be large (see [29]). Having defined this polynomial expansion we proceed to approximate the integrals in Eq.

5.26, thus leading to an approximation to Eq. 5.25 of the form:

$$\tilde{F}_N(\theta) = \sum_{|\mathbf{i}|=0}^{N} \tilde{a}_i \phi_{\mathbf{i}}(\theta), \tag{5.28}$$

$$\tilde{a}_{\mathbf{i}} = \sum_{j=1}^{Q} F(z) \phi_{\mathbf{i}}{}^{(j)}(\theta) \pi(\theta^{(j)}) w^{(j)}, \tag{5.29}$$

where $w^{(j)}$, with $j = 1, \ldots, Q$ are the weights corresponding to the Q quadrature points for a given integration rule. Clearly,

$$\tilde{F}_N(\theta) \to F(\theta) \text{ as } Q, N \to \infty. \tag{5.30}$$

Thus, we can use these methods in order to approximate the posterior. Such approximation is given by

$$\pi(\theta|\mathbf{d}) = \frac{\tilde{\pi}_N(\mathbf{d}|\theta)\pi(\theta)}{\int \tilde{\pi}_N(\mathbf{d}|\theta)\pi(\theta)d\theta}. \tag{5.31}$$

Thus, once an accurate gPC solution for $\tilde{F}_N(\theta)$ is obtained, the dependence on θ is known analytically, hence we would be able to sample from the posterior at a negligible cost.

Chapter 6

Numerical Examples

In this chapter we present experiments that illustrate Chapter 4-5 applied to seismic source inversion problems in one and two dimensions

6.1 A 1D Numerical Test

We generate synthetic data by solving the acoustic wave equation with given by

$$u_{tt} - u_{xx} = f(x,t), \quad (x,t) \in [-5.1, 5] \times [0, 3] \tag{6.1}$$

$$u(-5,t) = u(5,t) \tag{6.2}$$

$$f(x,t) = \delta'(x - x_0)2(h^2)(t-1)e^{-(w^2)(t-1)^2} \quad , \tag{6.3}$$

where we chose the true parameters to be $x_0 = 0, w = 6, h = 6$, where x_0 is the location of the source and w and h are proportional to the frequency and amplitude respectively. Note that this problem can be solved analytically and its solution is given by

$$u(x,t) = \begin{cases} 36(x-(t-1))e^{-36(x-(t-1))^2}, & x > 0 \\ 36(x+(t-1))e^{-36(x+(t-1))^2} & x \leq 0. \end{cases} \tag{6.4}$$

Thus, we have a closed form that we can use to generate our "recorded" data. Moreover, we know what the true values of our parameters are and as such we can use them to verify the validity of the results.

6.1.1 Deterministic Approach

We proceed to compare the methods described in Chapter 4. We recorded at 11 equally spaced receivers during 21 different times t_i, $0 < t_1 < \cdots < t_i < \cdots < t_{N_t} = 3$. For both deterministic examples we consider the convergence criteria to be $|\theta_{k+1} - \theta_k| < 10^{-5}$. The results are given in the following table, where $\theta = (x_0, w, h)$, θ_0 is the initial guess, k is the number of iterations, θ^* is the true value of the parameters, θ_k is the computed vector of parameters at iteration k and the error is given by the difference between the recorded data and the data generated by θ_k.

$\theta^* = (0, 6, 6), \quad \theta_0 = (1, 4, 7)$			
Mehtod	θ_k	Error	k
Steepest Descent	(-2.582(-2) 5.999, 5.9993)	2.5849(-2)	17
CG	(3.0267(-3), 6.0094, 6.016)	1.723(-2)	9
BFGS	(0.2185, 6.001, 5.9847)	2.191(-2)	8

$\theta^* = (0, 6, 6), \quad \theta_0 = (1, 7, 7)$			
Mehtod	θ_k	Error	k
Steepest Descent	(-2.58(-2), 5.99, 5.999)	2.4364(-2)	17
CG	(8.013(-3), 6.003, 5.982)	1.920(-2)	7
BFGS	(2.925(-2), 6.0681, 6.0150)	3.001(-2)	4

As we can see from the tables, these methods are quite sensitive to the choice of initial point. Moreover, we can see that, in general BFGS converges faster than CG. This comes as no surprise because BFGS makes use of the Hessian, or at least an approximation to it, which in turn provides more information about the search direction.

6.1.2 Probabilistic Approach

We next illustrate the Bayesian approach by considering an array of 11 receivers collecting data at 21 different time instances between $t = 0$ and $t = 6$. The vector of parameters θ is given by $\theta = (x_0, w, h, \sigma^2) = (0, 6, 6, 0.3)$.

Calibration Model I: Metropolis-Hastings

We start by considering a calibration model which uses the closed form solution as a forward model. We perform the MCMC sampling by using the standard Metropolis-Hastings algorithm explained in Chapter 5. The choice of priors was given by

$$\begin{cases} x_0 \sim U(-4, 4), \\ w \sim U(4, 7), \\ h \sim U(5, 8), \\ \sigma^2 \sim \Gamma^{-1}(1, 1), \end{cases} \tag{6.5}$$

where the ranges for the priors are determined by an estimation of the physical parameters of the model (i.e "expert information"). We ran the MCMC algorithm for 50,000 iterations with a burn-in period of 10,000. This in turns yields the posterior distribution of the parameters shown in Figure 6.1.

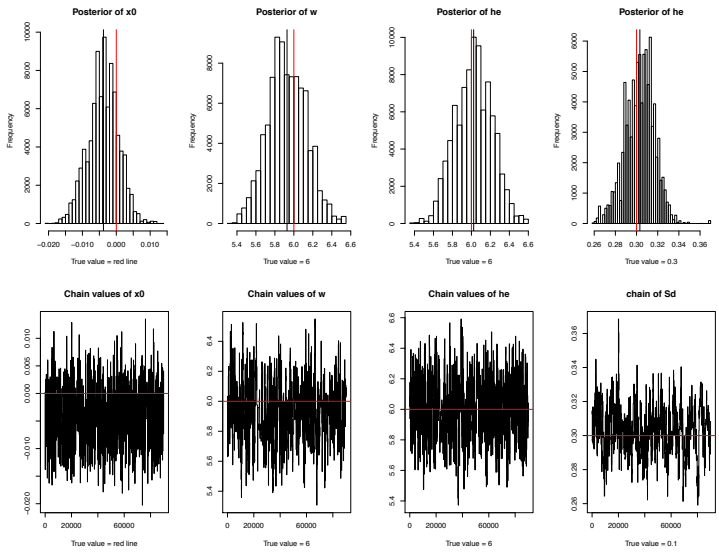

FIGURE 6.1: Posteriors for the priors given by Eq. (6.5).

As we can see in Figure 6.1, the chains seem to mix appropriately. Moreover,

the probability density function of the posterior captures the true values of the parameters. A table of the characteristics of the distribution is given by

	x_0	w	h	σ^2
Min.	-0.0321353	5.306	4.514	0.2591
1^{st} Quantile.	-0.0065644	5.7871	5.884	0.2943
Median	-0.0036877	5.910	6.023	0.3047
Mean	-0.0036479	5.924	6.019	0.3040
3^{rd}	-0.0007988	6.071	6.159	0.3131
Max.	0.6209674	6.549	6.592	0.4235

From the previous table we can see that the average values are indeed very close to the true values of the parameters, which validates the model.

Another experiment included the following priors:

$$\begin{cases} x_0 \sim U(-3,3), \\ w \sim N(5,1), \\ h \sim U(5,8), \\ \frac{1}{\sigma^2} \sim \Gamma(1,1/10). \end{cases} \tag{6.6}$$

And yields the posterior distributions shown in Figure 6.2. Note that in this case the posterior distributions of h, w, and σ^2 are not able to capture the true value of the parameter, however, they get relatively close to it. The posterior distributions for x_0 is able to capture this true value.

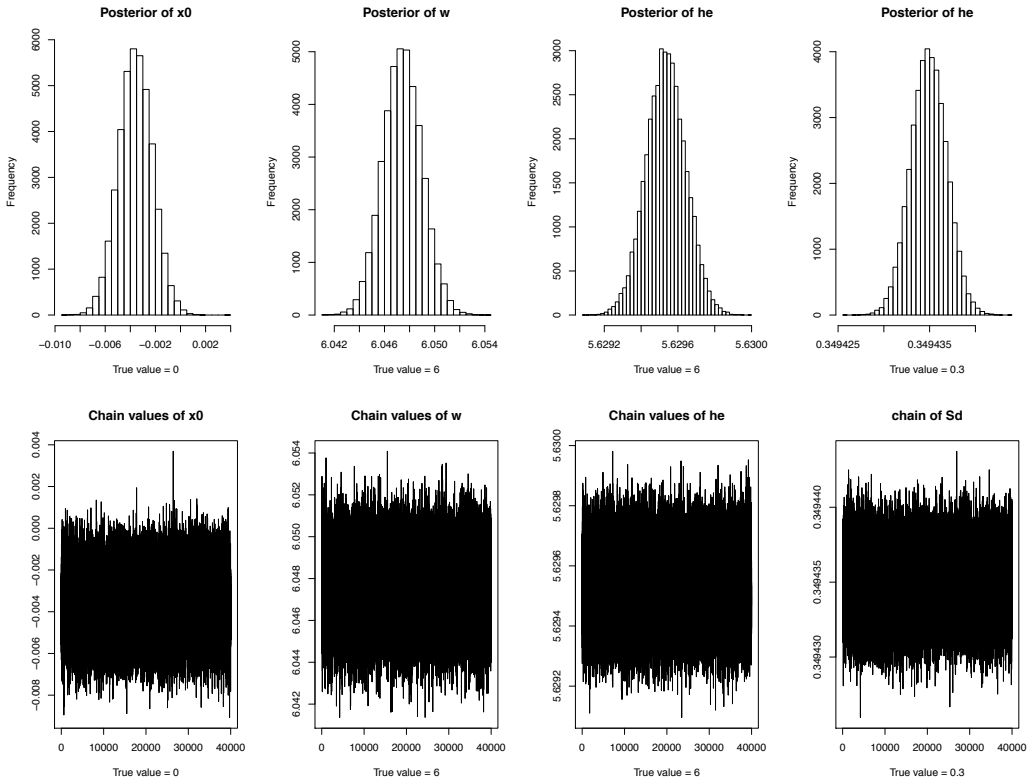

FIGURE 6.2

A third attempt uses

$$\begin{cases} x_0 \sim U(-1,1), \\ w \sim N(5,1), \\ h \sim N(7,1), \\ \sigma^2 \sim \Gamma^{-1}(1,1), \end{cases} \tag{6.7}$$

and yields the posterior distributions shown in Figure 6.3, where an issue similar to the one presented in Figure 6.2 arises.

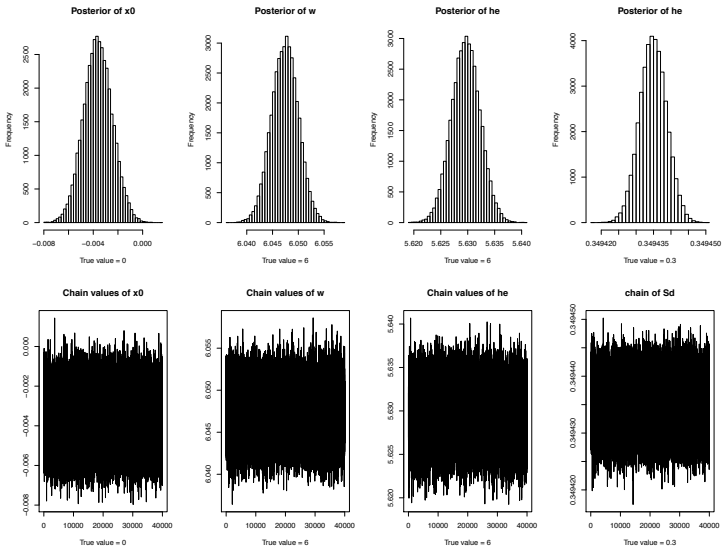

FIGURE 6.3

From the three experiments shown, the most accurate results are obtained using the priors given by Eq. (6.5).

Forward Model:

We now proceed to perform MCMC algorithm using our forward discontinuous Galerkin model. In order to produce a fair comparison, we use the same priors presented in Eq. (6.5). Doing so we get the following distributions for the posteriors.

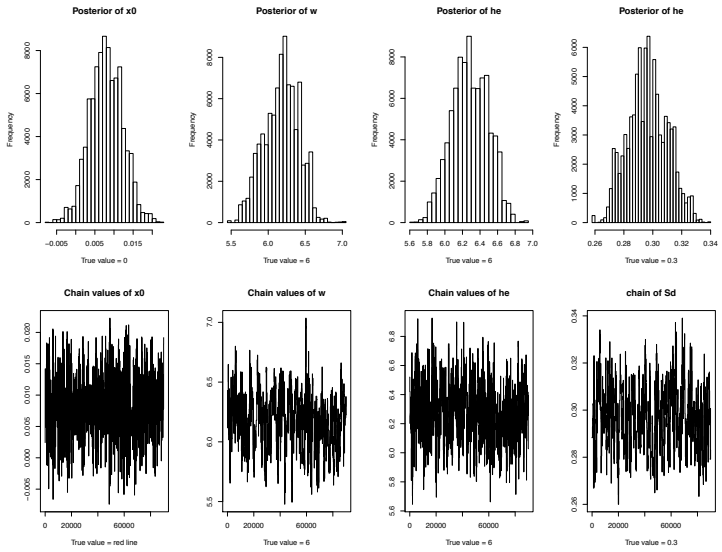

FIGURE 6.4: Bayesian Inversion using dG

As we can see from Figure 6.4, we obtain similar posterior distributions to the ones presented in Figure 6.1. Note that the true values of the parameter are captured in the posterior distributions.

6.1.3 Experiment 3: Comparison Between Serial and Parallel MCMC

We now proceed to compare how the serial and parallel MCMC algorithm compare. Selecting the same priors as in Eq. (6.5), we proceed to perform the inversion both for the closed form solution and the Forward model. The results are given by Figure 6.5. The total number of iterations was 5,000, on which the parallel MCMC divided them into 4 processors. The running times were as follows:

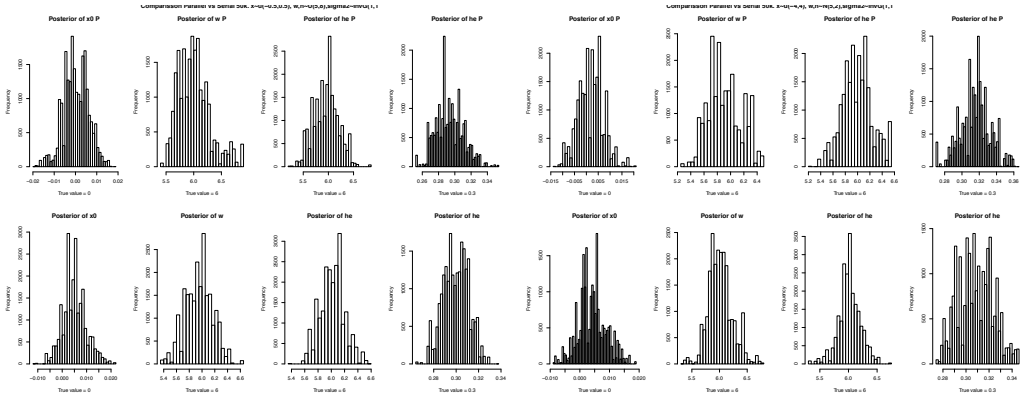

FIGURE 6.5: Comparison between Parallel MCMC and serial MCMC. On the left we have the results for parallelizing the MCMC using the closed form solution. To the right we have the results for when the forward model is used. The top graphs are the ones for the parallelized code and the bottom ones represent the serial results.

As we can see, the distributions seem behave similarly. Moreover, we are able to capture the true values of the parameters, which deems this method as useful, A comparison of the running times is given in Table (6.1).

	Exact Model	dG Model
Serial MH	172.4 s	1787.5 s
Parallel MH	60.7 s	640.6s
Gain	2.84	2.79

TABLE 6.1: Comparison between parallel and serial MCMC

As we can see, we obtain a performance gain of about 3 times. In theory, this gain should be a number closer to 4 times. We believe the reason behind this is that this experiment was not run on a dedicated machine that had to run other processes at the same time. Regardless, the experiment shows how both methods compare and provides an experimental efficiency gain. Clearly, this gain would be higher if the method is implemented in a machine with more processors. In

order to compare the parallelized and the serial MH we utilize the Gelman-Rubin diagnostic plot, show in Figure 6.6.

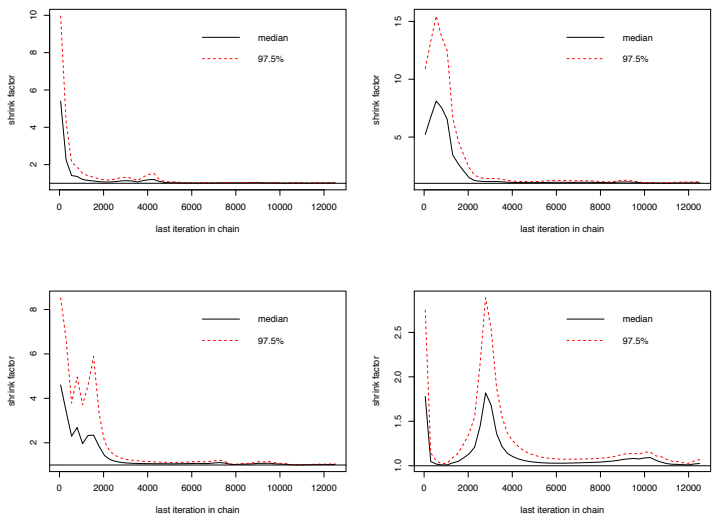

FIGURE 6.6: Gelman-Rubin diagnostic plot. At the top we have the diagnostic for the parameters x_0 and w_0, from left to right. At the bottom we have the diagnostic for h and σ^2.

As we can see from the plot, there is convergence, which implies that the parallelized method resembles the serial method. However, the plot also suggest that we should use a larger number of iterations in order to adequately capture the posterior distribution of σ^2.

6.2 A 2D Numerical Test

We now consider the two-dimensional case. This set of examples present higher complexity; not only we need to find more parameters, we also have to deal with a more complex forward model, which translates into a higher computational time. In order to overcome this difficulty, we can reduce the polynomial degree for the dG solver and simulate over a smaller time. However, the trade-off is that the

solver would be more imprecise, and as such the errors associated to the results might be somewhat high (although yet acceptable). Moreover, we do not count with the appropriate resources at the moment to run such demanding scheme hundreds or even thousands of times, so we have to resort to this approach..

We consider the two-dimensional acoustic wave equation with periodic boundary conditions given by

$$u_{tt} - \nabla^2 u = f(\mathbf{x}, t), \quad (\mathbf{x}, t) \in [0,1]^2 \times [0,2], \tag{6.8}$$

$$\tag{6.9}$$

with homogeneous Neumann boundary conditions

$$\nabla u \cdot \mathbf{n} = 0, \tag{6.10}$$

initial conditions given by

$$u(\mathbf{x_0}, 0) = 0, \quad u_t(\mathbf{x_0}, 0) = 0 \tag{6.11}$$

and source term given by

$$f(\mathbf{x}, t) = \mathbf{M}^T \nabla \delta(\mathbf{x} - \mathbf{x_0}) s(t), \tag{6.12}$$

$$s(t) = -2(w_0)^2 (t - t_0) e^{-(w_0(t-t_0))^2}, \quad \mathbf{M} = (a_1, a_2). \tag{6.13}$$

Note that in this case we have more parameter to invert, namely $x_0, y_0, t_0, a_1, a_2, w_0$, as opposed to just three parameters for the one dimensional case. We proceed as in Section 6.1 by first considering the deterministic inversion.

6.2.1 2D: Deterministic Approach

Just as in the previous section, we proceed to implement the inversion using the steepest descent, CG, and BFGS methods. We try them at different initial guesses and analyze their behavior. On each experiment we take θ^* as the true parameter. We consider 3 measuring devices located at $y = 1$ and throughout the x component of the domain. We denote steepest descent by SD and conjugate gradient by CG:

$\theta^* = (0.5, 0.5, 1, 5, 10, 20)$, $\theta_0 = (0.9, 0.9, 5, 7, 13, 23)$			
Mehtod	θ_k	Error	k
SD	(0.501, 0.502, 0.999, 5.016, 10.016, 20.025)	3.3(-2)	37
CG	(0.500, 0.500, 0.999,4.999,10.002,20.001)	1.2(-2)	27
BFGS	(0.498, 0.497, 0.996, 4.993, 9.993, 19.990)	1.5(-2)	22

$\theta^* = (0.5, 0.5, 1, 5, 10, 20)$, $\theta_0 = (0.1, 0.8, 3, 13, 8, 18)$			
Mehtod	θ_k	Error	k
SD	(0.505, 0.505,0.999, 5.010, 10.021, 20.054)	5.9(-2)	34
CG	(0.496, 0.499, 0.999, 4.999, 10.002, 20.001)	4.9(-2)	25
BFGS	(0.502, 0.503, 0.996, 4.997, 9.993, 19.990)	1.3(-2)	19

Thus, we can see that with a properly chosen initial guess we are able to obtain a vector of parameters very similar to the true one.

6.2.2 2D Probabilistic Case

We now consider the probabilistic approach to the 2D source inversion. We consider the same parameters as before, as well as an array of 5 receivers at $y = 1$ and

different locations in x. Given that the forward model is quite computationally expensive, we were only able to run each simulation a *handful* (∼5000) times, which in turn produced less than satisfactory results. We start by generating the "recorded" data using the following parameters: $\theta^* = (x_0, y_0, t_0, w_0, a_1, a_2, \sigma^2) = (0.1, 0.9, 0.1, 50, 1, 2, 0.2)$. We also test different combinations of priors and proposals. The following results were some of the most meaningful, where each experiment was run with 5000 iterations. We begin by using flat priors for all parameters except σ^2 and a normal (hence symmetric) proposal distribution. The priors are given by:

$$\begin{cases} x_0 \sim U(0, 0.3), \\ y_0 \sim U(0.2, 0.99), \\ t_0 \sim U(0, 0.3), \\ w_0 \sim U(40, 60), \\ a_1 \sim U(-3, 3), \\ a_2 \sim U(-4, 4), \\ \sigma^2 \sim \Gamma^{-1}(10, 2), \end{cases} \tag{6.14}$$

and the results are shown in Figure 6.7.

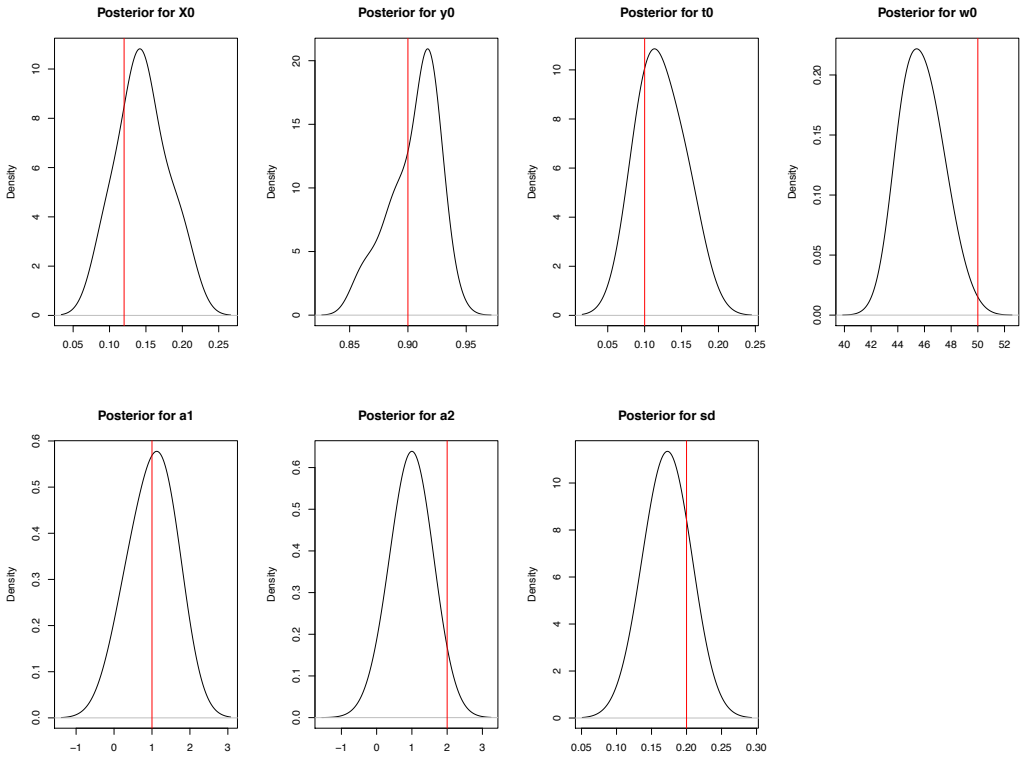

FIGURE 6.7: Results obtained using all flat priors and a normal proposal.

As we can see, we are able to capture the true values of our distribution with this choice of priors and proposal, however, the obtained distribution for the parameter w_0 puts the true value of w_0 on one of the tails of the distribution, i.e, it barely captures said value.

On a second attempt, we use again the same flat priors, however, our proposal function follows a Gamma distribution. The results are shown in Figure 6.8.

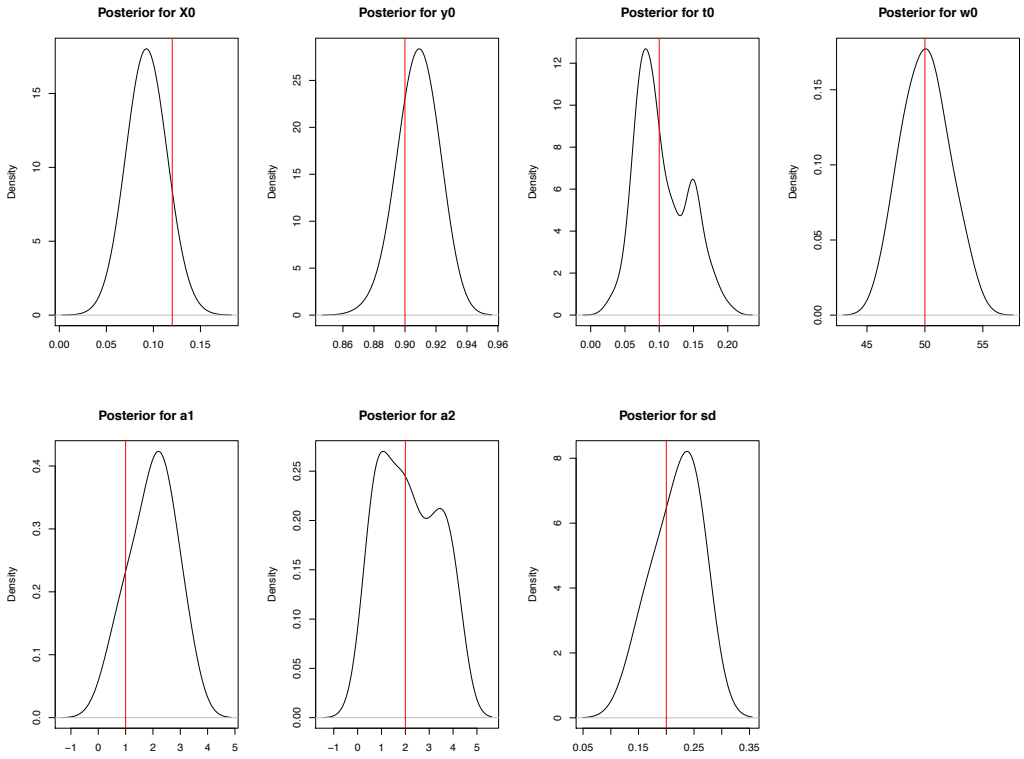

FIGURE 6.8

In this case we get to capture all the true values on a region with higher probability, however, contrary to Figure 6.7, the density curves are not as smooth.

Lastly, consider the following priors,

$$\begin{cases} x_0 \sim U(0, 0.2), \\ y_0 \sim U(0.2, 0.99), \\ t_0 \sim \Gamma(2, 20), \\ w_0 \sim \Gamma(102, 2), \\ a_1 \sim U(-3, 3), \\ a_2 \sim U(-4, 4) \\ \sigma^2 \sim \Gamma^{-1}(10, 2). \end{cases} \tag{6.15}$$

Using a Γ proposal distribution, we get the following results:

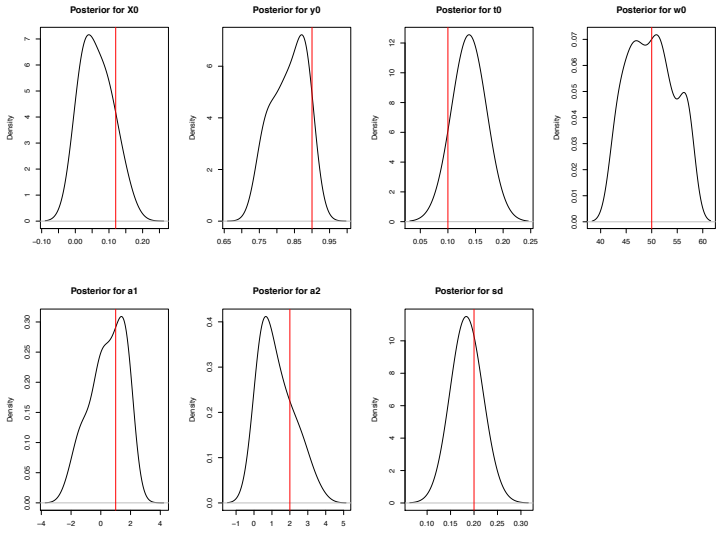

FIGURE 6.9: Just as with Figure 6.8, we are able to capture the true values of the parameters, however, we get less smooth posteriors.

We consider that the best choice of proposal distribution is a normal distribution with an appropriately chosen standard deviation. This means a standard deviation that is neither too small, so the whole support can be explored, or to

big so that the acceptance rate is not too small. In general, an acceptance rate of 25% is ideal, according to [30].

Lastly, we implement a parallelized version of the Metropolis-Hastings algorithm in order to solve the 2D inversion problem. We run the MH algorithm 5,000 times on 4 different processors. The proposal is given by a normal distribution and the priors are presented in Eq. (6.14). The results are shown in Figure 6.10.

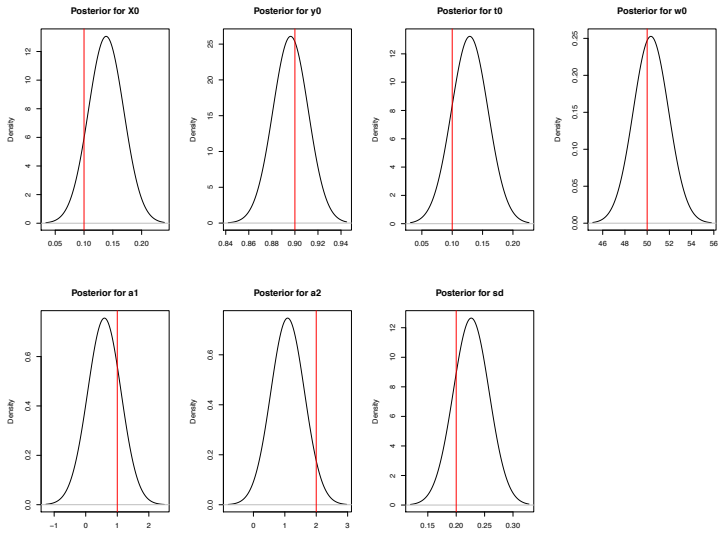

FIGURE 6.10: Results for the parallelized MH.

As we can see from Figure 6.10, each posterior is able to capture the true value of the parameters.

We conclude that the results obtained are not ideal, however they do provide some insight in the theory of MCMC. In this case we used a particularly expensive forward model, which effectively limited the number of iterations done. However, note that when the parallelization is used in order to increase the number of samples, we obtain smoother posterior distributions.

Chapter 7

Final Remarks

7.1 Discussion

We studied the seismic source inversion by deterministic and probabilistic approaches. Both methods can clearly be implemented in order to solve this source inversion problem however, they do have their advantages and disadvantages. we proceed to discuss them.

For the deterministic case, we were able to obtain satisfactory results using few number of iterations on the optimization algorithm. For our experiments, we usually need less than a hundred forward solves. This is desirable because, as stated earlier, the efficacy of the inversion is governed by the numerical efficiency of the forward model. However, this methods have some issues as well. Firstly, they often suffer from spurious minima and as such, they tend to be dependent on the initial guess used. Additionally, they will usually require the construction of a regularization function. Moreover, deterministic methods depend on at least a computation of the gradient of the cost functional with respect to the unknown parameters, which can be done by taking a finite difference approach for each parameter, however, this requires an evaluation of the forward model for each

component of the gradient. This was not too much of an issue in our case since we only had 3 and 6 components for the one and two dimensional case, however, this can clearly become an issue when more complicated models that include a larger amount of parameters are considered. One way to fix this issue is to compute the gradient using adjoint state methods. Finally, a subtle issue that arises with the deterministic approach is that it does not take into consideration the uncertainty in the model.

As for the probabilistic approach, we used a method that not only provides a reliable, flexible, and easy to implement way of performing a source inversion, but also takes into account the uncertainty in the model. Although both probabilistic and deterministic methods are fundamentally different and as such not very suitable to comparison between them, we consider that the probabilistic approach provides more information. Compared to the deterministic approach, the Bayesian framework provides a more *natural* way of quantifying our a priori knowledge. However, as mentioned before, The issue with these methods is that they require a large number of forward solves, which again is feasible for one dimension but gets more complicated as the number of dimensions increases.

Regardless of the issues that can arise with both methods, we consider that all implementations provided insightful results.

We would like to remind the reader that inverse problems is a rather large field in mathematics, engineering and science and as such these methods can be implemented to many other different inversion problems, not just seismic source inversion.

7.2 Future Work

Improving the efficiency of the methods is of paramount importance. There are some variations of the methods here presented that were not implemented and that would be interesting to do so since they provide a more efficient probabilistic inversion. Particularly the use of more advanced MCMC algorithms, such as Hamiltonian Monte Carlo. Additionally, we would like to further study and implement some of the more *sophisticated* methods, including gPC-based Bayesian inversion. Techniques aimed to improve the efficiency of the deterministic approach, such as the use of adjoint state methods and the implementation of the Wasserstein metric, are expected to be implemented in the future. As a longer term goal, the creation and implementation of novel methods to perform Bayesian inversion is expected as part of a doctoral dissertation.

Bibliography

[1] D. Appelö and T. Hagström, "A new discontinuous galerkin formulation for wave equations in second-order form", *SIAM Journal on Numerical Analysis*, vol. 53, no. 6, pp. 2705–2726, 2015.

[2] C. Burstedde and O. Ghattas, "Algorithmic strategies for full waveform inversion: 1d experiments", *Geophysics*, vol. 74, no. 6, WCC37, 2009, ISSN: 00168033. DOI: 10.1190/1.3237116.

[3] P. Mora, "Nonlinear two-dimensional elastic inversion of multioffset seismic data", *Geophysics*, vol. 52, no. 9, pp. 1211–1228, 1987.

[4] R. L. Mackie and T. R. Madden, "Three-dimensional magnetotelluric inversion using conjugate gradients", *Geophysical Journal International*, vol. 115, no. 1, pp. 215–229, 1993.

[5] A. Guitton and W. W. Symes, "Robust inversion of seismic data using the huber norm", *Geophysics*, vol. 68, no. 4, pp. 1310–1319, 2003.

[6] Y.-H. Liu and C.-C. Yin, "3d inversion for frequency-domain hem data", *Chinese Journal of Geophysics*, vol. 56, no. 12, pp. 4278–4287, 2013.

[7] J. Tromp, C. Tape, and Q. Liu, "Seismic tomography, adjoint methods, time reversal and banana-doughnut kernels", *Geophysical Journal International*, vol. 160, no. 1, pp. 195–216, 2005.

[8] B. Sjögreen and N. A. Petersson, "Source estimation by full wave form inversion", *Journal of Scientific Computing*, vol. 59, no. 1, pp. 247–276, 2014.

[9] J. Dettmer, S. E. Dosso, and C. W. Holland, "Model selection and bayesian inference for high-resolution seabed reflection inversion", *The Journal of the Acoustical Society of America*, vol. 125, no. 2, pp. 706–716, 2009.

[10] S. Stähler and K Sigloch, "Fully probabilistic seismic source inversion-part 1: Efficient parameterisation", *Solid Earth*, vol. 5, no. 2, p. 1055, 2014.

[11] B. Calderhead, "A general construction for parallelizing metropolis- hastings algorithms", *Proceedings of the National Academy of Sciences*, vol. 111, no. 49, pp. 17 408–17 413, 2014.

[12] Y. Marzouk and D. Xiu, "A stochastic collocation approach to bayesian inference in inverse problems", 2009.

[13] J. Martin, L. C. Wilcox, C. Burstedde, and O. Ghattas, "A stochastic newton mcmc method for large-scale statistical inverse problems with application to seismic inversion", *SIAM Journal on Scientific Computing*, vol. 34, no. 3, A1460–A1487, 2012.

[14] E. T. Chung and B. Engquist, "Optimal discontinuous galerkin methods for the acoustic wave equation in higher dimensions", *SIAM Journal on Numerical Analysis*, vol. 47, no. 5, pp. 3820–3848, 2009.

[15] M. Käser and M. Dumbser, "An arbitrary high-order discontinuous galerkin method for elastic waves on unstructured meshes?i. the two-dimensional isotropic case with external source terms", *Geophysical Journal International*, vol. 166, no. 2, pp. 855–877, 2006.

[16] L. C. Wilcox, G. Stadler, C. Burstedde, and O. Ghattas, "A high-order discontinuous galerkin method for wave propagation through coupled elastic–acoustic media", *Journal of Computational Physics*, vol. 229, no. 24, pp. 9373–9396, 2010.

[17] J. S. Hesthaven and T. Warburton, *Nodal discontinuous Galerkin methods: Algorithms, analysis, and applications*. Springer Science & Business Media, 2007.

[18] J. Nocedal and S. J. Wright, "Numerical optimization", *Springerverlang, USA*, 1999.

[19] I. Griva, S. G. Nash, and A. Sofer, *Linear and nonlinear optimization*. Siam, 2009.

[20] J. M. Papakonstantinou, *Historical Development of the BFGS Secant Method and Its Characterization Properties*. ProQuest, 2009.

[21] Q. Long, M. Motamed, and R. Tempone, "Fast bayesian optimal experimental design for seismic source inversion", *Computer Methods in Applied Mechanics and Engineering*, vol. 291, pp. 123–145, 2015, ISSN: 00457825. DOI: 10.1016/j.cma.2015.03.021. arXiv: 1502.07873.

[22] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines", *The journal of chemical physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[23] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications", *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[24] S. Chib and E. Greenberg, "Understanding the metropolis-hastings algorithm", *The american statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[25] W. Neiswanger, C. Wang, and E. Xing, "Asymptotically exact, embarrassingly parallel mcmc", *ArXiv preprint arXiv:1311.4780*, 2013.

[26] A. Gelman and D. B. Rubin, "Inference from iterative simulation using multiple sequences", *Statistical science*, pp. 457–472, 1992.

[27] S. P. Brooks and A. Gelman, "General methods for monitoring convergence of iterative simulations", *Journal of computational and graphical statistics*, vol. 7, no. 4, pp. 434–455, 1998.

[28] S. Brooks, A. Gelman, G. L. Jones, and X.-L. Meng, *Handbook of markov chain monte carlo*. Chapman and Hall/CRC, 2011.

[29] M. Motamed, F. Nobile, and R. Tempone, "A stochastic collocation method for the second order wave equation with a discontinuous random speed", *Numerische Mathematik*, vol. 123, no. 3, pp. 493–536, 2013.

[30] C. Robert and G. Casella, *Introducing Monte Carlo Methods with R*. Springer Science & Business Media, 2009.