ACOUSTIC AND ELASTIC WAVEFORM INVERSION BEST PRACTICES

Ryan T. Modrak

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Geosciences
Adviser: Professor Jeroen Tromp

January 2017
© Copyright by Ryan T. Modrak, 2017.

All rights reserved.
Abstract

Reaching the global minimum of a waveform misfit function requires careful choices about the nonlinear optimization, preconditioning and regularization methods underlying an inversion. Because waveform inversion problems are susceptible to erratic convergence, one or two test cases are not enough to reliably inform such decisions. We identify best practices instead using two global, one regional and four near-surface acoustic test problems. To obtain meaningful quantitative comparisons, we carry out hundreds acoustic inversions, varying one aspect of the implementation at a time.

Comparing nonlinear optimization algorithms, we find that L-BFGS provides computational savings over nonlinear conjugate gradient methods in a wide variety of test cases. Comparing preconditioners, we show that a new diagonal scaling derived from the adjoint of the forward operator provides better performance than two conventional preconditioning schemes. Comparing regularization strategies, we find that projection, convolution, Tikhonov regularization, and total variation regularization are effective in different contexts. Besides these issues, reliability and efficiency in waveform inversion depend on close numerical attention and care. Implementation details have a strong effect on computational cost, regardless of the chosen material parameterization or nonlinear optimization algorithm.

Building on the acoustic inversion results, we carry out elastic experiments with four test problems, three objective functions, and four material parameterizations. The choice of parameterization for isotropic elastic media is found to be more complicated than previous studies suggests, with “wavespeed-like” parameters performing well with phase-based objective functions and Lamé parameters performing well with amplitude-based objective functions. Reliability and efficiency can be even harder to achieve in transversely isotropic elastic inversions because rotation angle parameters describing fast-axis direction are difficult to recover. Using Voigt or Chen-Tromp pa-
rameters avoids the need to include rotation angles explicitly and provides an effective strategy for anisotropic inversion.

The need for flexible and portable workflow management tools for seismic inversion also poses a major challenge. In a final chapter, the software used to carry out the above experiments is described and instructions for reproducing experimental results are given.
Acknowledgements

It has been a pleasure and an honor to work with Yang Luo, Yanhua Yuan, Dmitry Borisov, Hom Nath Gharti, Matthieu Lefebvre, Wenjie Lei, Youyi Ruan, Heru Rusmanugroho and James Smith, with whom I shared extremely enjoyable research collaborations; Etienne Bachmann, Pathikrit Bhattacharya, Umair bin Waheed, Ebru Bozdağ, Shanna Chu, Leah Langer, David Luet, Gian Matharu, Daniel Peter, Frederik Simons, Uno Vaaland and Hejun Zhu, who provided many interesting and informative discussions; and Jeroen Tromp, who created an environment for research, growth and collaboration that I look back on with much gratitude and appreciation.
To my parents.
Contents

Abstract ............................................................... iii
Acknowledgements ..................................................... v
Preface ............................................................................ x

1 Introduction ............................................................... 1

2 Waveform inversion best practices: acoustic test cases 4
   2.1 Introduction .......................................................... 4
   2.2 Test problems ........................................................ 5
   2.3 Testing procedures .................................................... 7
   2.4 Nonlinear optimization algorithms ......................... 8
   2.5 Line search algorithms ............................................ 12
   2.6 Preconditioning ....................................................... 15
   2.7 Regularization ........................................................ 23
   2.8 Other considerations ............................................... 30
   2.9 Conclusions ........................................................... 33

3 On the choice of material parameters for isotropic elastic inversion 53
   3.1 Theory ................................................................. 55
   3.2 Test problems ......................................................... 56
   3.3 Testing procedures ................................................... 58
   3.4 Results ................................................................. 60
<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5</td>
<td>Conclusions</td>
<td>61</td>
</tr>
<tr>
<td>4</td>
<td>Anisotropic imaging with fast recovery of tilt and azimuth angles</td>
<td>66</td>
</tr>
<tr>
<td>4.1</td>
<td>Theory</td>
<td>68</td>
</tr>
<tr>
<td>4.2</td>
<td>Imaging experiments</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>Anisotropic inversion</td>
<td>78</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>78</td>
</tr>
<tr>
<td>5.2</td>
<td>Sensitivity analysis</td>
<td>88</td>
</tr>
<tr>
<td>5.3</td>
<td>Methodology</td>
<td>92</td>
</tr>
<tr>
<td>5.4</td>
<td>Numerical experiments</td>
<td>93</td>
</tr>
<tr>
<td>5.5</td>
<td>Conclusions</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>SeisFlows: a flexible and portable waveform inversion package</td>
<td>122</td>
</tr>
<tr>
<td>6.1</td>
<td>Introduction</td>
<td>122</td>
</tr>
<tr>
<td>6.2</td>
<td>Design goals</td>
<td>124</td>
</tr>
<tr>
<td>6.3</td>
<td>Package organization</td>
<td>126</td>
</tr>
<tr>
<td>6.4</td>
<td>High-performance computing example</td>
<td>135</td>
</tr>
<tr>
<td>6.5</td>
<td>Usage overview</td>
<td>141</td>
</tr>
<tr>
<td>7</td>
<td>Conclusions</td>
<td>148</td>
</tr>
<tr>
<td>A</td>
<td>Implementation Details</td>
<td>150</td>
</tr>
<tr>
<td>A.1</td>
<td>Limited-memory BFGS algorithm</td>
<td>150</td>
</tr>
<tr>
<td>A.2</td>
<td>Preconditioned nonlinear conjugate gradient method</td>
<td>151</td>
</tr>
<tr>
<td>A.3</td>
<td>Restart conditions</td>
<td>152</td>
</tr>
<tr>
<td>A.4</td>
<td>Line search termination conditions</td>
<td>153</td>
</tr>
<tr>
<td>B</td>
<td>Reproducibility</td>
<td>155</td>
</tr>
<tr>
<td>B.1</td>
<td>Checkerboard example</td>
<td>156</td>
</tr>
</tbody>
</table>
Preface

This dissertation consists of seven chapters, including an introduction, conclusion and five other chapters which are either publications, peer-reviewed conference proceedings or manuscripts about to be submitted. Chapters 2, 3 and 6 are mainly Ryan Modrak’s individual work under the supervision of Jeroen Tromp. Chapters 4 and 5 are from a collaboration in which Heru Rusmanugroho was the lead author and provided the forward modeling code and Ryan Modrak provided the inversion code. Yanhua Yuan proposed the phase-based misfit function and Dmitry Borisov performed the envelope-based inversion that played important parts in chapters 3 and 6, respectively.
Chapter 1

Introduction

Waveform inversion is a powerful technique for estimating Earth’s internal properties and imaging its discontinuities (Virieux and Operto, 2009). Unlike ray-based methods (Bording et al., 1987; Docherty, 1991), waveform inversion involves few or no approximations or simplifying assumptions in the mathematical treatment of wave propagation.

Historically, computational expense has been a major factor limiting the adoption of waveform-based techniques. Despite early statements of the theory (Loewenthal et al., 1976; Tarantola, 1984), the first three-dimensional proof-of-concept demonstrations awaited the development of “tera-scale” supercomputers (Komatitsch and Tromp, 2001). Mainstream adoption lagged behind even further, with two-dimensional inversions coming into widespread use only about two decades ago and three-dimensional inversions only about one decade ago. Even today, expense remains a significant limiting factor for 3D elastic near-surface applications, and 3D elastic continental- and global-scale waveform inversion has become feasible only with the arrival of “peta-scale” supercomputers (Zhu et al., 2012).

Computational cost in waveform inversion can be attacked on two broad fronts: through the forward problem and the inverse problem. Compiler optimization and
hardware acceleration can provide considerable savings in the forward problem (Peter et al., 2011). Cost savings in the inverse problem, on the other hand, can be achieved through algorithmic choices that reduce the number of model updates. In practice, the most aggressive model update strategies are not always the most cost-effective, since strategies that provide fast convergence at the beginning of an inversion often lead to an incorrect final models (Bunks et al., 1995).

In the literature for both oil and gas exploration and solid earth geophysics, many strategies have been proposed to provide efficiency without sacrificing robustness. Choices for the nonlinear optimization procedure, which drives the model updates by minimizing misfit between data and synthetics, include (in order of increasing complexity) nonlinear conjugate gradient methods, quasi-Newton methods and truncated Newton methods. Choices for the preconditioner include diagonal or quasi-Newton scalings. Choices for regularization include smoothing, projection and Tikhonov or total variation penalty functions. Elastic inversion requires additional decisions involving the material parameterization. Because formulas relating one set of elastic moduli to another are in general nonlinear, the choice of material parameterization can not bring about stretching or shrinking of the misfit surface, but also affect the degree of convexity and with it the ultimate success or failure of an inversion.

To help guide practitioners in these choices, this thesis employs a “brute force” approach for comparing model update strategies. Hundreds of inversions based on 2D acoustic and elastic regional, global and near-surface test problems are used to develop intuition and identify best practices. Chosen for applicability to more expensive 3D scenarios, these test cases provide a window into to body wave and surface wave inversion at various spacial scales.

This thesis follows a logical progression from acoustic to isotropic elastic and finally to transversely anisotropic elastic inversion. Chapter 2 examines nonlinear optimization, preconditioning and regularization issues in the acoustic case using
regional, global and near-surface test problems. Because computational cost and numerical ill-conditioning increase in moving from acoustic to elastic parameterizations, the cost-saving measures and numerical safeguards identified from the acoustic test problems become even more important in the elastic case. Chapter 3 examines the choice of material parameterization for isotropic elastic media. Chapters 4 and 5 extend the analysis to transversely anisotropic media first in the context of noniterative migration and then in the context of iterative inversion.
Chapter 2

Waveform inversion best practices: acoustic test cases

2.1 Introduction

Waveform inversion practitioners must choose from a variety of objective functions, nonlinear optimization algorithms, preconditioning strategies, regularization methods, and multiscale schemes. Though an extensive applied mathematics literature exists on these topics, much of it is based on numerical benchmarks that are less challenging and computationally expensive than waveform inversion problems. In the waveform inversion literature itself, methodological comparisons sometimes lack implementation details, involve only one or two test cases, or use starting models quite close to the global minimum of the objective function.

To address such issues, we provide systematic comparisons between inversion strategies through six acoustic inversion test cases. While a comparison of objective functions and multiscale procedures would fit naturally into this framework, we choose to focus instead on numerical aspects of inversion—nonlinear optimization,
preconditioning and regularization—that have received somewhat less attention in the geophysical literature.

Robustness and efficiency in waveform inversion depend in large part on numerical decisions. Because of their importance, such choices ought to be informed by both practical waveform inversion experience and numerical theory and results. Benchmark comparisons by Nash and Nocedal (1991) and Zou et al. (1993) and review papers by Nocedal (1992), Gould et al. (2005), and Burstedde and Ghattas (2009) supply a useful foothold into the numerical literature. The emerging field of PDE-constrained optimization (Biegler et al., 2003)—involving parameter estimation, optimal design, and optimal control systems governed by partial differential equations—offers additional relevant experience. Through wide-ranging references, we seek to ground the waveform inversion results below in this literature.

This paper is organized as follows. Section 2 and 3 provide a description of test cases and testing procedures. Sections 4 and 5 present the results of a comparison of nonlinear optimization algorithms, focusing first on the search direction and then on the line search. Sections 6 and 7 discuss preconditioning and regularization. A number of other issues that do not fit well into any of the previous categories are covered in section 8. Finally, in section 9, we conclude with a list of seismic waveform inversion “best practices”.

### 2.2 Test problems

We present convergence results from the following test cases: (a) Marmousi, (b) over-thrust, (c) salt, (d) anticline, (e) global, (f) regional, and (g) deep Earth. Through target models representing horizontal or vertical cross sections, each of these 2D problems provides a window into an associated 3D problem.
Problems a–d correspond to widely used exploration geophysics test cases. True models, shown in Figure 1, include various thrust fault, normal fault, and salt structures. Smooth starting models were obtained by convolving true models with Gaussian kernels. Inversions were carried out in the acoustic approximation, that is, using acoustic models and data to approximate the elastic subsurface. For the overthrust and Marmousi test cases, we considered both onshore and offshore variants to give a sense for how performance differs between these two cases.

Problems e–g investigate seismic inversion at much larger scales. Wavefield simulations are once again based on the acoustic wave equation, with an analogy to horizontal surface wave propagation for the regional and global test cases and vertical compressional wave propagation for the deep Earth test case. For the regional and global problems, starting models were homogeneous, and for the deep Earth problem the starting model was a radial reference model, AK135. The true model for the global test case was based on 40 s Rayleigh wave phase speeds from Trampert and Woodhouse (2003). The true model for the regional test case was based on 10 s Rayleigh wave phase speeds from Ekström et al. (2009). Finally, the true model for the deep Earth test case was obtained by superimposing Gaussian random variations on AK135.

For the global test problem, periodic boundary conditions were used at the sides of a rectangular mesh to roughly approximate the spherical Earth. For the deep Earth test problems, the inner and outer core were included in wavefield simulations, but excluded from model updates.

For the near-surface problems a–d, data from 32 shots were simulated at 500 hydrophones. Shots and hydrophones were placed at 10 m depth in a 500 m water layer. Multiple reflections were excluded from both data and synthetics.

For the regional and global problems e–g, sources and receivers were chosen to mimic the actual distribution of earthquakes and seismic stations on Earth’s surface.
For the global test case, sources were constrained to plate boundaries and receivers to dry land. For the deep Earth test case, stations were constrained to Earth’s surface and earthquakes to < 300 km depth.

### 2.3 Testing procedures

Inversions were performed in the time domain using a waveform difference objective function,

\[
\chi(m) = \frac{1}{2} \sum \int |s(m, t) - d(t)|^2 dt ,
\]  

(2.1)

where \(m\) is the model, \(s\) are synthetics, \(d\) are observations, and the sum is over all sources and receivers. For simplicity, no muting or windowing of traces was performed in any of the inversions.

In describing algorithms, we sometimes write the model, gradient, and Hessian as scalar functions of spatial position. Generalization to the multi-parameter case, we note, is usually just a matter of introducing a sum over material properties. Because forward modeling dominates computational expense, cost comparisons are made in terms of wavefield simulations. In displaying convergence results, we plot \(L2\) model error \(\int |m - m_{\text{true}}|^2 dV\), where the integral is over spatial position, versus the cumulative number of wavefield simulations. We recall that each model update requires at least two sets of wavefield simulations, one for the gradient and one for the line search. The gradient evaluation itself, strictly speaking, requires two sets of wavefield simulations, but the second contributes no new cost to the inversion since it is carried over from the previous line search.

For forward and adjoint simulations, we used the spectral element solver **SPECFEM2D** (Komatitsch and Vilotte, 1998b), which employs an explicit time stepping scheme and an “optimize-then-discretize” approach to the adjoint operator (Gunzburger, 2000). For nonlinear optimization, data preprocessing, gradient post-
processing, and workflow integration tasks we used the SeisFlows framework. Both are open source packages available through GitHub.

2.4 Nonlinear optimization algorithms

The rate of convergence in waveform inversion depends on the nonlinear optimization algorithm used to iteratively update the model. The work of a model update, conventionally, is divided into two steps. First, a search direction is computed based on the gradient of the objective function. Second, a step length is determined along the search direction through a line search procedure. In this section, we compare two widely used search direction algorithms, leaving detailed discussion of the line search until section 5.

2.4.1 Limited-memory BFGS algorithm

The L-BFGS algorithm (Liu and Nocedal, 1989) is a quasi-Newton method, which means that search directions are based on a low-dimensional quadratic model of the objective function. After several decades of experience with such methods, L-BFGS is generally regarded as the most effective quasi-Newton method (Nocedal, 1992; Kolda et al., 1998).

Over the course of an inversion, the quadratic model of the objective function formed by L-BFGS varies through an updating process, with each new gradient evaluation providing more information. L-BFGS is limited-memory in the sense that results from only the most recent gradient evaluations need to be stored. L-BFGS search directions are well-scaled in the sense that they terminate at the vertex of the paraboloid used to locally represent the objective function. For reference, a concise statement of the L-BFGS algorithm is given in appendix A.
To specify the number of gradient evaluations kept track of by L-BFGS, users must choose a memory value. For waveform inversion problems, we find that values between three and seven work well. Generally, it seems that problems with high nonlinearity benefit from lower memory values, and problems with low nonlinearity benefit from higher memory values, though differences are often quite minor. Supporting results are provided in the online supplement.

2.4.2 Nonlinear conjugate gradient method

The nonlinear conjugate gradient method (NLCG) returns a search direction that is a linear combination of the gradient and the previous search direction. Among several NLCG variants, those due to Polak and Ribièrè (1969) and Gilbert and Nocedal (1992) have proven particularly effective. Because NLCG does not involve a quadratic model of the objective function, the length of the search direction is not especially meaningful and, hence, more effort must be expended on the line search. For reference, a concise statement of the NLCG algorithm is given in appendix B.

Although NLCG and L-BFGS share certain theoretical underpinnings (Nazareth, 1979), the two algorithms provide quite different user experiences. While the L-BFGS search direction computation is a more complicated than the NLCG search direction computation, L-BFGS may be easier to implement overall, perhaps, on account of its simpler initial step length selection, line search, and restart procedures. Importantly, both algorithms can be combined with stochastic inversion strategies for additional savings (van Leeuwen et al., 2011; van Leeuwen and Herrmann, 2013; Castellanos et al., 2015).

As an aside, we note that linear and nonlinear conjugate gradient methods differ in that while the former are used to solve systems of linear equations, the latter are used to solve nonquadratic optimization problems. Here we consider only nonlinear conjugate gradient methods, noting in passing that the use of linear conjugate gradient
methods to solve a series of linear subproblems forms the basis for another nonlinear optimization algorithm, the truncated Newton method (Nash, 2000), which comprises an active research area (Métiévier et al., 2014; Burstedde and Ghattas, 2009).

2.4.3 Comparisons

L-BFGS has been shown to provide computational savings over NLCG and other competitors in a number of classic studies (Liu and Nocedal, 1989; Nash and Nocedal, 1991; Zou et al., 1993; Kolda et al., 1998). Many of these early comparisons involved inexpensive, low-dimensional optimization problems from a list compiled by Moré et al. (1981). As Gould et al. (2005) point out, more recent benchmarks are in short supply.

There are significant differences between early nonlinear optimization test cases and waveform inversion problems not only in terms of computational expense and model space dimensionality, but also in terms of nonlinearity and nonconvexity. A set of updated performance comparisons might therefore be useful. To provide one such benchmark, we compared the efficiency of L-BFGS and NLCG in experiments with the waveform inversion examples described above. Out of curiosity, the steepest descent algorithm was also included in the comparisons. To allow straightforward comparisons between test problems, we used the “regularization by convolution” method described in section 6, Polak-Ribiére NLCG search directions, and an L-BFGS memory value of five in all cases. The results of these experiments, shown in 2.2, show L-BFGS as the clear winner, with computational savings of 30 to 50 percent over NLCG.

These findings provide the starting point for much further analysis. In section 4, we describe how a backtracking line search procedure contributes to the efficiency of L-BFGS. In section 5 we demonstrate additional computational savings through preconditioning. Finally, in section 6, we show how regularization can help solve convergence problems evident in the nonlinear optimization comparisons.
Having used the waveform inversion test cases to compare optimization algorithms, we can looking at things the other way around, use the results to infer how nonlinearity varies from one test case to another. The most obvious differences occur between near-surface problems and regional and global problems. While the convergence rate in the near-surface inversions is often slow and erratic, the regional and global inversions settle quickly into superlinear convergence. Among the near-surface problems themselves there is considerable variation, with the overthrust models, which generate strong diving waves and weak reflections, converging fastest, and the Marmousi models, which generate weak diving waves and strong reflections, converging slowest.

2.4.4 Numerical issues

Nonlinearity in waveform inversion can cause significant problems for optimization algorithms. Drawing on the waveform inversion test cases, we describe two numerical problems that can usually be resolved by restarting, that is, by discarding the algorithm’s accumulated state and continuing as if no prior gradient evaluations were available. More details about restart conditions are given in appendix C.

Lack of a descent direction

The most basic requirement of a search direction is that it provides a reduction in the objective function. If \( p \) is the search direction and \( g \) is the gradient, then \( p \) must satisfy

\[
p^T g < 0
\]

(2.2)

in order to provide such a decrease. Explicitly checking equation this condition adds virtually no cost to an inversion since it requires only a vector product.

In waveform inversion, occasional failure of the optimization algorithm to provide a descent direction is not unexpected. Out of some 1800 model updates carried out for
Figure 2.2 L-BFGS and NLCG required restarts about one percent of the time. The restart rate can be much higher, we find, in applications involving multiparameter inversion, noisy data, or stochastic optimization, though we do not include any such test cases here.

**Loss of conjugacy**

On highly nonlinear problems, NLCG search directions gradually lose the property of conjugacy, or orthogonality with respect to an inner product involving the Hessian, on which good performance of the method depends. Fast convergence can usually be regained by restarting the algorithm, as described for example by Powell (1977). In our experience, such safeguards are occasionally necessary in waveform inversion to avoid stagnation.

### 2.5 Line search algorithms

Given a model $m$ and search direction $p$, the work of the line search is to find a step length $\alpha$ such that the updated model $m + \alpha p$ satisfies the decrease and curvature conditions described in appendix D. In section 4, we compared search direction algorithms using the type of line search appropriate for each one: for NLCG we used a bracketing line search, and for L-BFGS we used a safeguarded backtracking line search. We now give an idea of the issues involved with both search procedures, and through numerical experiments examine their contribution to the overall cost of an inversion.

#### 2.5.1 Bracketing line search

Acceptable NLCG step lengths can vary by several orders of magnitude from one model update to another. During the line search, a good strategy to deal with this
lack of scaling is to first bracket the minimum of the objective function along the search direction and then choose a step length by polynomial interpolation between bracketing points.

Since the bracketing procedure can add considerably to the cost of an inversion, it must be carried out efficiently. To avoid unnecessary gradient evaluations, we check the curvature condition only after the minimum has been bracketed and a polynomial interpolation has been performed. After this, if a given step length is found to satisfy both descent and curvature conditions, the associated gradient evaluation can be carried over to the next model update iteration, removing the need for any additional evaluations until the next line search.

### 2.5.2 Safeguarded backtracking line search

The L-BFGS algorithm with proper initialization returns a search direction that is well-scaled in the sense that a unit step length $\alpha = 1$ is an appropriate first choice. While most of the time a unit step satisfies the decrease condition mentioned above, occasionally one or more subsequent trial steps are required. The idea of a backtracking line search is to select any subsequent trial steps by interpolating backward, towards zero, within the unit interval. Because the search direction has to satisfy equation (2.2), a reduction in the objective function relative to $\alpha = 0$ can always be found in the unit interval. Our use of the term “safeguarded” relates to the fact that if the backtracking procedure fails to return a step length satisfying the curvature condition, we terminate the backtracking line search and switch to a bracketing line search.

For choosing backtracking step lengths, we use the quadratic and cubic interpolation algorithms given by Nocedal and Wright (2006a). To ensure that the interpolation procedure does not select a step length too close to zero on the one hand or too close to the old step length on the other, we impose upper and lower bounds of
the type described by Dennis and Schnabel (1996). Since well-scaled L-BFGS search directions are available only after at least two gradient evaluations have been performed, we switch from a bracketing line search to a backtracking line search starting with the second model update iteration.

2.5.3 Comparisons

Figure 2.3 shows results from numerical experiments involving the line search. In the waveform inversion test cases, a typical bracketing line search is found to require 3.5 function evaluations, and a typical backtracking line search is found to require 1.2 function evaluations.

Importantly, in waveform inversion and other optimization procedures based on adjoint methods, the last function evaluation of the line search overlaps with the first function evaluation of the next model update iteration. Put another way, a forward simulation performed during the line search removes the need for a forward simulation later on as a prerequisite for the next adjoint simulation. As a result, a backtracking line search contributes little to the overall cost of an inversion, only about 0.2 function evaluations on average. At about 2.5 function evaluations, the effective cost of a bracketing line search in the waveform inversion is significantly higher. In the numerical optimization literature, Nash and Nocedal (1991) reported a similar cost per L-BFGS backtracking line search. Published cost estimates for bracketing line searches vary more widely, reflecting the greater diversity of algorithms in use. Liu and Nocedal (1989) report an average cost of around 2.5 function evaluations per NLCG bracketing line search, less expensive than in the waveform test problems. We note that Figure 2.3 of this paper presents essentially the same method comparisons as table 13 of Liu and Nocedal, though with expensive, high-dimensional waveform inversion problems considered in this study and comparatively inexpensive, low-dimensional test problems considered in the other.
2.6 Preconditioning

The performance of the nonlinear optimization algorithms discussed above can be improved through preconditioning. In this section, we begin with a general overview and move on to descriptions and numerical comparisons of waveform inversion diagonal preconditioners.

2.6.1 Overview

Preconditioning is a way of rescaling or recombining model parameters to provide favorable numerical properties. Despite upfront computational and storage costs, such a procedure can provide significant overall savings by accelerating the convergence of the optimization algorithm.

Underlying most preconditioning methods is a change of variables via a linear transformation, say \( \hat{m} = Cm \). In preconditioning conjugate gradient methods, the change of variables \( C \) does not enter the computations directly, except through the action of \( P = C^TC \) or its inverse. \( P \) is typically called the preconditioner, even though its inverse is usually what is implemented in practice. Importantly, a good tradeoff between inexpensive computation of \( P \) and fast convergence of the optimization algorithm is required for the preconditioner to provide an overall reduction in computational cost.

Typically, the term “preconditioning” is used in connection with conjugate gradient methods and “rescaling” in connection with quasi-Newton algorithms. While preconditioning can be used to connote a general change variables, aimed at providing any numerical property that accelerates convergence, rescaling is more often thought of in terms of an initial approximation to the Hessian.

Most preconditioning strategies involve direct or indirect connections to the Hessian. Even for computationally demanding problems, the Hessian can be made to
play a useful role through numerical approximations. Given the huge dimensionality of the model space in waveform inversion, it is important that such approximations allow for inexpensive computation and affordable storage. Examples of this kind of approach include the following.

1. Quasi-Newton preconditioners, in which the approximation to the Hessian varies from one model update to another through an updating process.

2. Higher-dimensional but still relatively inexpensive preconditioners involving, for example, forward simulations with coarse numerical grids or approximate solvers.

3. Diagonal preconditioners formed by exact or inexact computation of the diagonal elements of the Hessian.

While the first two categories have received some attention (Akcelik et al., 2003; M´etivier et al., 2014; Demanet et al., 2010), most preconditioners in waveform inversion fall into the last category (Claerbout and Nichols, 1994; Shin et al., 2001; Rickett, 2003). In suggesting best practices below, we focus on diagonal preconditioners not because they are always the most cost effective, but because they are widely used, easily implemented, and can serve as a starting point for more sophisticated techniques.

2.6.2 Waveform inversion preconditioners

Two types of diagonal preconditioners are prevalent in waveform inversion: scalings that account geometric spreading away from the sources, and scalings obtained by applying the adjoint of the forward solver to the data (Rickett, 2003). Using perturbation analysis, Luo (2012) showed that both types of preconditioners are related, but not exactly equivalent, to second order variations of a waveform-difference misfit function. We now briefly restate Luo’s results.
By expanding the displacement field, $u$, as a function of the model, $m$, in a perturbation series

$$u(m + \delta m) \approx u(m) + \delta u_1(m) + \delta u_2(m), \quad (2.3)$$

the variation in waveform-difference misfit can be written as

$$\delta \chi \approx \delta \chi_0 + \delta \chi_1 + \delta \chi_2, \quad (2.4)$$

where $\delta \chi_1$ and $\delta \chi_2$ correspond to first- and second-order scattering terms $\delta u_1$ and $\delta u_2$. If $H_1$ is the positive semidefinite first-order contribution to the waveform-difference Hessian and $H_2$ is the remaining second-order contribution, the gradient $g$ and Hessian $H = H_1 + H_2$ are related to the variation of the data misfit via

$$\delta \chi_0 = \int g(x) \delta m(x) \, dV, \quad (2.5)$$

$$\delta \chi_1 = \frac{1}{2} \int \int \delta m(x) \, H_1(x, x') \, \delta m(x') \, dV \, dV', \quad (2.6)$$

$$\delta \chi_2 = \frac{1}{2} \int \int \delta m(x) \, H_2(x, x') \, \delta m(x') \, dV \, dV'. \quad (2.7)$$

By taking

$$\lim_{x \to x'} H(x, x') \quad (2.8)$$

various diagonal scalings can be derived. Referring to Luo (2012) for details, we state the main result of the perturbation analysis, namely, that diagonal preconditioners

$$P_1(x) = \sum_{i=1}^{N_s} \int \partial_t^2 u(x, t) \, \partial_t^2 u(x, t) \, dt \quad (2.9)$$

$$P_2(x) = \sum_{i=1}^{N_s} \int \partial_t^2 u(x, t) \, \partial_t^2 v(x, -t) \, dt \quad (2.10)$$
are related to the data misfit variations (2.6) and (2.7), respectively, through the limit (2.8). If $G$ is the Green’s function of the medium, then in the above expressions

$$u(x, t) = \int G(x, s, t - t') f(t') \, dt'$$

(2.11)

is the wavefield originating from the source located at $s$ with wavelet $f(t)$, and

$$v(x, t) = \sum_{j=1}^{N_r} \int G(x, r_j, t - t') \left[ d_j(t') - u(r_j, t') \right] \, dt'$$

(2.12)

is the data residual wavefield that arises from backprojecting the differences between observed data $d_j(t)$ and simulated data $u(r_j, t)$ from the source located at $s$ and the receivers located at $r_j$, $j = 1, \ldots, N_r$.

While both $P_1$ and $P_2$ contribute to the variation in data misfit through the diagonal of the Hessian, each behaves in a different way and it is useful to consider them separately. $P_1$ involves only the wavefield originating from the sources, so it does a better job than $P_2$ accounting for amplitude effects such a geometric spreading, focusing, and defocusing. $P_2$ involves wavefields originating from both the sources and receivers, so it is more effective than $P_1$ in compensating for uneven data coverage. Because $P_1$ and $P_2$ correspond to the first- and second- order contributions the Hessian, respectively, it sensible to precondition using either $P_1$ or $P_1 + P_2$ but not $P_2$ alone. We show later that $P_1$ and $P_1 + P_2$ provide almost identical performance.

Given their direct relation to the waveform-difference Hessian, $P_1$ and $P_2$ can be viewed as variations on a theme developed by exploration geophysicists first from the perspective of migration, and later from the perspective of waveform inversion. In migration, Rickett (2003) compared the performance of a source-only diagonal scaling, similar to $P_1$, with diagonal scalings derived to the adjoint of the forward solver, similar to $P_2$. 
Meanwhile, in regional and global seismology, the existence of well-behaved crust, mantle, and core phases led to the adoption of wave-equation based phase and traveltime misfit functions (Dahlen et al., 2000; Tromp et al., 2005). Such methods combine the robustness of full waveform modeling with the reduced nonlinearity of phase and traveltime measurements compared with amplitude and waveform-difference measurements. A preconditioner that arises naturally in this context is

\[ P_3(x) = \sum_{i=1}^{N_s} \int \partial_t^2 u(x, t) w(x, -t) \, dt, \]  

(2.13)

where

\[ w(x, t) = \sum_{j=1}^{N_r} \int G(x, r_j, t - t') \partial_t u(r_j, t') \, dt'. \]  

(2.14)

In the terminology of Marquering et al. (1999), \( P_3 \) is simply the unweighted sum of “banana-doughnut kernels” from all source-receiver pairs.

The similarity between this and the other two preconditioners suggests the possibility of using \( P_3 \) as an alternative diagonal approximation to the waveform-difference Hessian. As shown through the checkerboard example below, \( P_3 \) has properties that make it appear quite promising.

The main work of generating \( P_1 \), \( P_2 \), and \( P_3 \), we note, consists of propagating the forward wavefields \( u \) and the reverse wavefields \( v \) and \( w \) from equations (2.9)–(2.13). For illustration, Figure 2.4 shows these preconditioners computed using a checkerboard test case. To drive \( v \), data residuals were obtained by subtracting traces generated from the checkerboard model shown in panel (a) with traces generated from a homogeneous model. The wavefields \( u \), \( v \), and \( w \) themselves were propagated within the same homogeneous model.

In panels (b) and (c) of Figure 2.4, we consider two source-receiver distributions. The first consists of a single source-receiver pair; the corresponding \( P_1 \) has a radially symmetric pattern, and \( P_2 \) and \( P_3 \) have alternating positive and negative fringes.
typical of wave-equation sensitivity kernels (Woodward, 1992; Dahlen et al., 2000). The second source-receiver distribution consists of 25 sources and 132 receivers in a typical regional seismology layout; in this case, $P_1$ has a noticeably smaller condition number, or “spread” of values than the other two preconditioners because it involves only wavefields originating from the sources.

Importantly, for the case of a single source-receiver pair, $P_2$ has more pronounced negative fringes than $P_3$. It follows that, for multiple source-receiver pairs, $P_2$ has a mix of positive and negative values and $P_3$ has mostly positive values. Given numerical problems associated negative eigenvalue with distributions (Fletcher, 1976), $P_3$ is expected to compare favorably to other preconditioners derived from the action of the adjoint of the forward solver.

### 2.6.3 Numerical issues

Even with diagonal approximations to the Hessian of the type illustrated in Figure 2.4, a number of nontrivial implementation questions remain. Here we focus two issues, smoothing and updating, with significant practical effects.

**Smoothing**

Away from a minimum or maximum of the objective function, the Hessian may have both positive and negative eigenvalues. As Figure 2.5 shows, the diagonal preconditioners $P_1 + P_2$ and $P_3$ may exhibit problematic eigenvalue distributions of this kind, which can cause numerical instability of the type described by Fletcher (1976). A simple and effective remedy, we find, is to replace negative values with zero and then smooth the resulting discontinuities.

Even if a diagonal preconditioner has no negative values, a large ratio of maximum to minimum values can result in slow or unstable convergence. While Rickett (2003) recommended damping to deal with such problems in migration, we recom-
mend smoothing in waveform inversion. Use of a damped preconditioner \( P + \lambda I \) works well in migration because it preserves the ability of the preconditioner to bring out fine details. In waveform inversion, it is desirable to bring out such details gradually to ensure that small-scale structures are not systematically mislocated as a result of errors in overlying large-scale structures. For this reason, smoothing the preconditioner, or some combination of smoothing and damping, works better than damping alone.

In practice, the amount of smoothing needed for good numerical performance is quite large. To investigate this issue quantitatively, we convolved preconditioners with Gaussian kernels with standard deviation \( \sigma = \sigma_x = \sigma_z \) measured in terms of grid spacing \( h \). Since grid spacing is related to dominant wavelength of the excited wavefield through a numerical condition, e.g., five grid points per wavelength, this way of looking at things is directly related to thinking about smoothing in terms of dominant wavelength. Comparing the performance of the resulting smoothed preconditioners in the waveform inversion test cases, we found that \( \sigma = 80h \) provides the best results on average, with somewhat lower smoothing required in the regional and global test cases and somewhat higher smoothing required in the near-surface test cases. A great deal of supporting information is provided in the online supplement.

Figure 2.7 gives a sense of the visual appearance of preconditioners after smoothing. For the near-surface test cases, preconditioners display both depth and lateral dependence, with shallow regions below the center of the array weighted differently than deep regions below the edges of the array. For the regional and global test cases, the lateral variations are even more pronounced as a result of uneven earthquake and seismic station distributions.

Though we do not perform such experiments here, we note that the convolution procedure described above can easily be modified to allow dip-dependent smoothing for near-surface problems or radial smoothing for regional and global problems. In
advocating smoothing, our motivation is purely numerical. Without it, preconditioning with the diagonal of the Hessian can result in slower than expected convergence, or even cause the inversion to fail. Interestingly, Symes (2008) describes a sense in which a filtering operation, similar to smoothing, provides a better underlying approximation to the Hessian.

**Updating**

Besides deciding how much to smooth, practitioners must choose how often to update a preconditioner to account for variations in the Hessian from one part of the model space to another. While updates improve the approximation to the Hessian, they also require forward and adjoint simulations whose cost might not be offset by faster convergence. Because updating the diagonal approximation Hessian amounts to variable preconditioning, it can also cause numerical difficulties of the type described by Knyazev and Lashuk (2007), which can be resolved by restarting the optimization procedure at the expense of slower convergence.

For diagonal preconditioners based on the waveform-difference Hessian, the question of how often to update depends on the relative size of the positive semidefinite first-order approximation $P_1$ and the remaining second-order contribution $P_2$. As shown in Figure 2.7, $P_1$ varies a little and $P_2$ varies a lot throughout the model space. From the numerical experiments below, we find it is not necessary to update either $P_1$, $P_1 + P_2$, or $P_3$ very often. In our experience, it is most effective to update diagonal preconditioners only at multiscale transitions, if at all.

**2.6.4 Comparisons**

After the smoothing procedure described above, we compared the numerical performance of preconditioners $P_1$, $P_1 + P_2$, and $P_3$ in the waveform inversion test cases. Figures 2.8 and 2.9 show the results of these experiments. In terms of convergence
rate, $P_1$ and $P_1 + P_2$ provide virtually identical performance. The new scaling $P_3$ provides computational savings, sometimes quite significant, relative to the other two. In Figures 2.9–2.12 we use the label “L-BFGS” for the case $D = I$, and “rescaled L-BFGS” for the case $D = P_3^{-1}$, with $D$ being the diagonal scaling described in appendix A. Likewise, we use the label “NLCG” for the case $P = I$ and “preconditioned NLCG” for the case $P = P_3$, with $P$ being the preconditioner described in appendix B.

### 2.7 Regularization

Without robust measures to mitigate nonconvexity and nonuniqueness in waveform inversion, the optimization algorithms and preconditioning strategies described above would be of little use. An important way of promoting convexity and suppressing nonuniqueness is to regularize, that is, to impose smoothness or other constraints on the model directly through the objective function or indirectly by other means.

Conceptually, regularization differs from preconditioning in a fundamental way: as a change of variables, preconditioning leaves the underlying optimization problem unchanged, while regularization, in effect, trades one problem for another more tractable problem (Engl et al., 2000).

Below, we describe several common regularization methods, discuss practical complications that arise during their use, and compare their performance through waveform inversion test cases. Although the following regularization methods are all to one degree or another classical, some interesting new facts emerge from analyzing them in the waveform inversion context.
2.7.1 Tikhonov regularization

In Tikhonov regularization, a preference for smooth models enters through the addition of a penalty term

\[ R_2(m) = \frac{\lambda}{2} \int_V \nabla m \cdot \nabla m \, dV, \quad (2.15) \]

to the objective function, where \( \lambda \) is a user-supplied parameter that controls the weight of the penalty term relative to the data misfit function. More commonly, the right hand side is written in discretized form as

\[ \frac{\lambda}{2} \sum_i \left[ (\partial_x m)^2_i + (\partial_z m)^2_i \right], \quad (2.16) \]

where the sum is over numerical grid points. Classic references on Tikhonov regularization include \cite{hansen1998discrete} and \cite{vogel2002computational}.

As an alternative to penalizing first-order spatial derivatives of the model, one can choose to work with higher-order spatial derivatives or some other measure of model roughness. In our experience, higher-order Tikhonov regularization often provides good results in regional and global inversions but not in near-surface problems.

Sometimes it makes sense to apply the penalty term not to the model itself, but to the difference between the model and some reference. We use such an approach for the deep Earth test case, applying the regularization term to the difference between the model and a radial reference model to avoid penalizing boundaries between the crust, mantle, and core.

Although the theory is classical, application of Tikhonov regularization to waveform inversion is not straightforward. The behavior of the penalty term can be quite different than in other seismic inverse problems, as illustrated in Figures 2.13 and C.11 through a checkerboard example. The use of a fine model discretization, as required for the solver computations, leads to abrupt variations in the derivatives of the penalty term. While the penalty term still acts to smooth the updated model,
it effectiveness is reduced. To work around this difficulty, it is possible to project
from a fine grid for the solver to a coarse grid for computing spatial derivatives of the
model and back again. The smoothness of the updated model in such an approach
derives from a combination of the regularization term and the projection operator.
Projection can also be used as a regularization method in its own right, as described
later in section 7.3.

### 2.7.2 Total variation regularization

Total variation (TV) regularization also acts directly through the objective function. The TV penalty term

\[
R_{1,\epsilon}(m) = \lambda \int_V \sqrt{\nabla m \cdot \nabla m + \epsilon} \, dV 
\]  

(2.17)

is essentially the $L_1$ norm of the spatial derivatives of the model. In discretized form, the right hand side becomes

\[
\lambda \sum_i \sqrt{(\partial_x m)^2_i + (\partial_z m)^2_i + \epsilon} .
\]  

(2.18)

Because the TV penalty terms acts through the $L_1$ norm, discontinuous transitions are not penalized any more than smooth transitions. As a result, the method is well-suited for recovering layered geologic structures.

While the damping parameter $\epsilon$ makes the above expressions differentiable, the effect of TV regularization on the gradient of the objective function is still problematic. To illustrate, Figure 2.13 shows the contribution to the gradient from the TV penalty term for different $\epsilon$ values. Even with large damping, abrupt variations in the derivatives of the penalty term remain that add considerably to the numerical difficulty of an inversion. Numerical problems associated with the TV regularization, it turns out, are common to a range of scientific applications (Vogel, 2002, Goldstein and Osher).
(2009) reviewed methods that have been proposed to make TV regularization numerically tractable. Recently, Lin and Huang (2015) applied one such workaround to acoustic and elastic waveform inversion with promising results.

2.7.3 Projection and convolution

Another way to apply regularization is through the basis functions used to represent the model (Engl et al., 2000; Mathé and Pereverzev, 2003). The choice of a smooth basis, for example, imposes a degree of smoothness on the model.

In waveform inversion with finite-difference or finite-element solvers, stability conditions are usually too strict for the numerical grid to suppress nonuniqueness. It becomes necessary, if the goal is to mitigate nonuniqueness through the choice of basis functions, to project back and forth from the fine grid used by the solver to a coarser basis used for the model update procedure. The projection can be performed either explicitly, or as described by Peters et al. (2015), implicitly by formulating the inversion as a constrained optimization problem. While the primary use of projection is as a regularization method, faster convergence may be a beneficial side effect since the efficiency of gradient-based optimization methods improves with decreasing model space dimensionality (Sigmund and Petersson, 1998).

A closely related method of regularization involves smoothing the gradient by convolving it with a Gaussian function or other kernel. While retaining mathematical properties of projection, convolution avoids the need to convert back and forth between two sets of basis functions, thus simplifying the inversion machinery. To see the connection between projection and convolution, consider a set of basis functions that differ only by spatial offset. Projection onto such a basis is equivalent to convolution with one of the basis elements. Convolution and other methods for smoothing the gradient have a long history in optimal control and design (Jameson, 1988), and a
more recent history in geophysical parameter estimation (Brenders and Pratt, 2007; Oh and Min, 2013; Alkhalifah, 2015).

2.7.4 Choice of regularization parameters

The extent to which regularization affects the result of a given model update can be adjusted through user-supplied parameters. In Tikhonov and TV regularization, the smoothness of the updated model is controlled in a straightforward way by the weight of the penalty term relative to the data misfit function. In projection and convolution, smoothness is a function of the spacing of the basis elements or the width of the convolution kernel. Such parameter choices are important for managing how quickly structure is added to the model and, in turn, for helping the updated model remain in the basin of convergence.

For testing the projection method of regularization, we explicitly convert back and forth between the fine solver basis and a coarser model update basis, using a set of spatially offset Gaussian functions as the elements of the latter. Relative to the solver grid spacing, we increase the spacing between Gaussian functions (as well as the the radius of each Gaussian function) by a uniform factor $f$, so that in the two-dimensional waveform inversion test cases the ratio $N$ of the number of basis functions to the number solver grid points is $N = f^{-2}$. For testing the convolution method of regularization, we use a Gaussian kernel to smooth the gradient prior to passing it to the search direction algorithm. The standard deviation $\sigma = \sigma_x = \sigma_z$ of the Gaussian function determines the amount smoothing.

To furnish values for the regularization method comparisons below, we adopt a “brute force” approach, running inversions multiple times to determine the most effective parameter values for a given test case. Results from these experiments are shown in the online supplement. Far from suggesting a routine method for parameter selection, the goal of these experiments is to build intuition and to ensure that in the
comparisons below, each regularization method performs at its best. Since regularization involves a tradeoff between fitting the data and suppressing nonuniqueness, increased convergence rate in one norm is often accompanied by decreased convergence rate in the other. As a result, decisions about which parameter value is best, in our approach, are made solely on the basis of model misfit. A number of less exhaustive, more practical methods for parameter selection are discussed by Vogel (2002).

2.7.5 Comparisons

Results from numerical experiments using the inversion test cases, regularization methods, and parameter selection procedure described above are shown in Figures 2.15–2.18.

Of all approaches, the performance of Tikhonov regularization is perhaps the most surprising. Since it acts through $L_2$ norm of the spatial derivatives of the model, Tikhonov regularization favors smooth transitions over discontinuous transitions in the model. Given this preference, it might be expected to perform well in recovering the regional model, which contains only smoothly varying structures, but this is not the case. Poor performance on the regional test case appears closely related to model discretization. Because the spatial derivatives of the model are computed using the fine numerical grid, Tikhonov regularization is much less effective than projection or convolution in smoothing the updated model. Increasing the weight on the regularization term does not reliably solve the problem because with gradient-based methods, the weight on the penalty term cannot exceed a certain value or the optimization algorithm becomes unstable.

While performing poorly on the regional example, Tikhonov regularization does well on all other test problems, especially the near-surface problems. The success of the method in these cases has a lot to do with improved accuracy at depth. By
smoothing layer interfaces, Tikhonov allows deep layers corresponding to reflected phases to shift position as shallow layers become more accurately recovered. Such a mechanism is important early in an inversion, when incomplete recovery of shallow structure leads to systematic errors in deep structure. Crucially, the smoothing effect is not large enough to prevent the emergence of layered structures, so updated models are able to generate reflections.

The results for TV regularization are also in some ways unexpected. Not surprisingly, TV regularization performs well in test cases involving discontinuous structures, especially the salt test case. What is surprising is that, despite its advantage in resolving layered structures, TV regularization performs worse than Tikhonov regularization in all near-surface test cases, which can be attributed to the well-known nonlinearity and ill-conditioning of the TV penalty function (Goldstein and Osher, 2009).

In most test cases, projection and convolution provide outcomes that, while similar to one another, differ markedly from Tikhonov and TV regularization results. In the regional test case, projection and convolution provide considerably better results than either penalty function method, with about a 2 times greater reduction in model error and $10^2$ times greater reduction in data misfit. Close inspection of figures 2.16 and 2.17 shows that convolution performs better than projection in the near-surface examples. The reason, we believe, is that while convolution allows small-scale structures to emerge over multiple model updates (even if no individual update contains such structures), projection to a coarser basis unavoidably limits recovery of such details. Close inspection of Figure 2.18 suggests that projection is more effective than convolution at suppressing nonuniqueness in regional and global inversions.

Looking at all test cases together, the importance of a problem-dependent perspective on regularization comes across strongly. While all four regularization methods are found to be useful in one way or another, clear differences emerge between
problems. Tikhonov regularization performs well when challenging small-scale structures are present, as in the near-surface problems. TV regularization provides slower convergence than Tikhonov regularization in most cases as a result of well-known numerical difficulties. Although we do not experiment with workarounds of the type described by Goldstein and Osher (2009), Lin and Huang (2015) showed that TV regularization can be successfully adapted to waveform inversion problems through such measures. Rather than simply being beneficial, projection or convolution may be all but required for dealing with highly uneven source-receiver distributions. Extrapolating from these results, we predict projection or convolution are most effective far away from the global minimum, Tikhonov regularization is effective closer to the global minimum, and TV regularization without numerical workarounds is only effective very close to the global minimum. In practice, use of two or more regularization methods at once may provide some advantages. A combination of Tikhonov and TV penalty terms in the objective function may be particularly beneficial, helping avoid problems associated with either method alone (Lin and Huang, 2015).

2.8 Other considerations

Next, we describe considerations that, although important to the success or efficiency of an inversion, do not fit well into any of the previous categories.

2.8.1 Multiscale transitions

In section 7, we compared regularization methods under controlled conditions, without varying the data misfit, data filtering, or regularization parameters from one model update to another. While good for building intuition, such an approach is not enough to get reliably to the global minimum of a waveform inversion objective
function. To avoid problems along the way, robust multiscale procedures are needed (Bunks et al., 1995).

Many multiscale procedures involve modification of the objective function. It may be beneficial to restart the optimization algorithm following such a change. By restarting, one avoids making comparisons in the NLCG and L-BFGS algorithms between the current and previous gradient, which may not be valid if the objective function has changed.

To determine best practices, we examined the performance of the L-BFGS algorithm in a two-level multiscale procedure with and without restarting. Modifying the objective function through the data filtering parameters, we carried out 25 model updates at low frequency and 50 model updates at high frequency. In terms of dominant frequency, the two multiscale levels differed by a factor of two.

The online supplement shows the performance of L-BFGS in these experiments. In all cases, restarting the algorithm at the transition between multiscale levels led to faster convergence. In three cases, the effect was relatively small (less than a one or two model update difference in computational cost), in two cases it was larger, and in the remaining four cases not restarting caused the optimization algorithm to fail outright.

2.8.2 Near field artifacts

From inaccuracy in the numerical treatment of wave propagation in the close vicinity of sources and receivers, the gradient of the data misfit function computed using the adjoint of the forward solver is commonly found to contain spurious near field features. Whether or not a correction is required depends on the regularization method employed. Convolution with a Gaussian kernel or projection onto a coarse basis tend to smooth out near field artifacts, so an additional correction is not generally required.
in these cases. Tikhonov or TV regularization, on the other hand, are not effective in smoothing out such features, so an correction is required in these cases.

To illustrate the problem, consider the checkerboard example in Figure 2.13. Panel (a) shows the true model, panel (b) the locations of sources and receivers, and panel (c) the gradient with respect to a homogenous initial model. Numerical artifacts around sources and receivers give the gradient a pockmarked appearance. In finite-element or finite-difference forward modeling, such artifacts can be removed by smoothing within a radius of one or two elements or grid points around each source and receiver. The use of this of type of procedure in waveform-difference inversion, it turns out, strongly parallels the use of source and receiver corrections of the type described by Tian et al. (2007) in traveltime inversion.

Let $x_i, i = 1, ..., N_r + N_s$ denote the location of the $i$-th source or receiver. To correct the raw gradient of the data misfit function, we compute for each $x_i$

$$
g_i = \int_V g_{raw}(x) \exp \left[ -\left( \frac{x - x_i}{h} \right)^2 \right] dV,
$$

using a quadrature rule that is appropriate for the given numerical discretization and a value $h$ that is one or two times the grid or element spacing. In subsequent computations, we use the corrected gradient given by

$$
g(x) = \frac{1}{N_r + N_s} \sum_{i=1}^{N_r+N_s} \left\{ g_{raw}(x) + [g_i - g_{raw}(x)] \exp \left[ -\left( \frac{x - x_i}{h} \right)^2 \right] \right\}.
$$

In Figure 2.13 the result of applying this correction to the raw gradient in panel (c) is shown in panel (d).
2.8.3 Masking strategies

Many inversions involve a water layer or other well-constrained region. If the properties of such an area are known with certainty, the corresponding model parameters can be excluded from the inversion. In other cases, it is better to include them in some way, for example, because there is uncertainty regarding the lower boundary of a salt structure or the properties of a water layer.

Bayesian methods, while arguably the most natural approach for incorporating such constraints, can add considerable complexity to an inversion. To find a simpler alternative, we compare two other strategies. The first involves modifying the NLCG preconditioner or the L-BFGS initial Hessian. In this approach, diagonal elements corresponding to well-constrained model parameters are scaled away from zero. The second strategy, which we call masking, involves ad hoc scaling of the gradient of the objective function. In this approach, gradient values corresponding to well-constrained model parameters are scaled toward zero.

We tested both methods using offshore exploration test cases. Masking the gradient performs better than rescaling the preconditioner, sometimes significantly better. By limiting changes to the water layer, both methods perform better than if no scaling is applied. Supporting results are provided in the online supplement. Although we experimented with water layers, the main usefulness of such methods, we anticipate, would be in dealing with salt structures.

2.9 Conclusions

Drawing on all of the results above, we suggest the following “best practices” for seismic waveform inversion.

For nonlinear optimization, we recommend L-BFGS over NLCG. Average savings of one third to one half in regional, global, and near-surface test cases make L-BFGS
the clear winner in terms of computational efficiency, on which the choice of one optimization algorithm over another primarily depends. L-BFGS provides other advantages as well, including the potential for a safeguarded backtracking search in place of a more complicated bracketing line search. Even with robust regularization and multiscale strategies, numerical problems can occur with both L-BFGS and NLCG. Restart conditions provide a way of detecting and recovering from such difficulties.

In place of diagonal scalings obtained by application of the adjoint operator to the data or to the difference between data and synthetics, we recommend a new scaling, which is shown to provide faster, more reliable convergence. To avoid numerical problems, it is necessary to smooth or damp diagonal preconditioners; we recommend smoothing as part of a strategy for bringing out details gradually over many model updates. The amount of smoothing required in practice can be quite large. Regularly updating preconditioners to account for variations in the Hessian from one part of the model space to another does not appear to be cost effective.

For regularization, we stress the importance of a problem dependent perspective. Tikhonov regularization performs well in near-surface test cases, the effect being large enough to promote convexity and suppress nonuniqueness, but not large enough to prevent recovery of layered structures. TV regularization suffers from well-known numerical difficulties, but workarounds have been developed that make the method competitive. Regularization by projection or convolution is often required for dealing with the types of source-receiver distributions and starting models that commonly occur in regional and global seismology.
Figure 2.1: Waveform inversion test cases (a) Marmousi, (b) overthrust, (c) salt, (d) anticline, (e) regional, (f) global, and (g) deep Earth. For the exploration test cases (a–d), target models are shown on the left and starting models on the right. For the regional and global test cases (e–g), target models are shown on the left and sources (magenta) and receivers (green) on the right. Because they are homogeneous, starting models for the regional and global test cases are not shown.
Figure 2.2: Comparison of nonlinear optimization algorithms. L-BFGS provides significant computational savings over NLCG in the waveform inversion test cases.
Figure 2.3: Comparison of line search algorithms. A safeguarded backtracking line search contributes significantly to the overall efficiency of L-BFGS. Because NLCG search directions are not well-scaled, a backtracking line search is not effective with NLCG, and a more expensive bracketing line search is required instead.
Figure 2.4: Illustration of waveform inversion diagonal preconditioners using a checkerboard example. *Top:* Experimental setup. *Middle:* Preconditioners corresponding to Network 1. *Bottom:* Preconditioners corresponding to Network 2.
Figure 2.5: With most diagonal preconditioners, smoothing or damping is required to avoid numerical problems. Damping helps bring out small structures quickly, which can be useful in non-iterative migration. Smoothing brings out such details gradually, helping avoid local minima in inversion. The smoothing parameter has significant effects on numerical performance, as discussed in the text. Top: Diagonal preconditioners before smoothing. Bottom: After smoothing.
Figure 2.6: How often is it necessary to update preconditioners? The answer depends on how much the Hessian varies throughout the model space. **Top:** Preconditioners computed using a homogeneous model. **Middle:** Preconditioners computed using a smoothed target model. **Bottom:** Preconditioners computed using an unsmoothed target model.
Figure 2.7: Left: Target model and/or network. Middle: Diagonal preconditioners $P_1 + P_2$. Right: Diagonal preconditioners $P_3$. 
Figure 2.8: Comparison of diagonal preconditioners. $P_1$ and $P_1 + P_2$ provide almost identical performance. $P_3$ performs better than the other two.
Figure 2.9: Comparison of optimization algorithms with and without preconditioning NLCG or rescaling the L-BFGS initial Hessian.
Figure 2.10: Comparison of nonlinear optimization algorithms. From top to bottom: Marmousi offshore, Marmousi onshore, and salt diapir inversion results after 100 function/gradient evaluations.
Figure 2.11: Comparison of nonlinear optimization algorithms. From top to bottom: overthrust offshore, overthrust onshore, and anticline inversion results after 50, 50, and 100 function/gradient evaluations, respectively. Because the overthrust inversions converge more quickly, results are shown after half the usual number of simulations.
Figure 2.12: Comparison of nonlinear optimization algorithms. From top to bottom: regional, global, and Deep Earth inversion results after 100 function/gradient evaluations. Because regional and global models lack the kind of coherent geologic structures found in the other test cases, we plot the difference between inverted and target models, rather than inverted models themselves.
Figure 2.13: Tikhonov and total variation regularization illustrated using a checkerboard example. (a) Target model. (b) Network. (c) Numerical mesh. (d) Gradient of data misfit function. (e) Gradient of data misfit function after applying source-receiver corrections described in section 8.2, which are essential for avoiding instability from the regularization penalty term. (f) Contribution to the gradient of the objective function from the Tikhonov penalty term. (g–i) Contribution to the gradient of the objective function from the total variation penalty term, with various choices of damping parameter $\lambda$. 
Figure 2.14: Behavior of regularization methods illustrated through a checkerboard example. Each panel above shows the inversion result after 5 model updates from a homogeneous starting model, with the regularization method and weight varied from one panel to another. Perhaps surprisingly, Tikhonov regularization is less effective than projection or convolution at smoothing the updated model. Numerical difficulties from the use of total variation regularization are apparent even at this early stage in the inversion.
Figure 2.15: Comparison of regularization methods. To ensure that each method performed at its best, we adopted a “brute force” approach to the selection of regularization parameters. The effectiveness of projection, convolution, Tikhonov, and total variation methods of regularization is found to be highly problem dependent.
Figure 2.16: Comparison of regularization methods. From top to bottom: Marmousi offshore, Marmousi onshore, and salt diapir inversion results after 50 function/gradient evaluations.
Figure 2.17: Comparison of regularization methods. From top to bottom: \textit{overthrust offshore}, \textit{overthrust onshore}, and \textit{anticline} inversion results after 50 function/gradient evaluations.
Figure 2.18: Comparison of regularization methods. From top to bottom: regional, global, and deep Earth inversion results after 50 function/gradient evaluations. Because regional and global models lack the kind coherent geologic structures found in the other test cases, we plot the difference between inverted and target models, rather than inverted models themselves.
Chapter 3

On the choice of material parameters for isotropic elastic inversion

The material parameterization used to compute derivatives, search directions, and model updates in elastic waveform inversion can have a significant effect on the robustness and efficiency of the overall nonlinear optimization procedure. For isotropic media, conventional wisdom holds that it is better to work with compressional- and shear-wave speeds $\alpha$ and $\beta$ than Lamé parameters $\lambda$ and $\mu$. Conventional wisdom further holds that, given their improved scaling and reduced covariance, bulk and shear moduli $\kappa$ and $\mu$, or “wave speed-like” parameters $\sqrt{\frac{\kappa}{\rho}}$ and $\sqrt{\frac{\mu}{\rho}}$, are an even better choice. Numerical tests involving hundreds of inversions, six subsurfaces models, and three full-waveform misfit functions, however, reveal a more complicated picture.

Working in the time domain and using numerically safeguarded quasi-Newton model updates, we find that the relative performance of material parameterizations is strongly problem dependent. Despite their physical relevance, $\alpha$ and $\beta$ rank last among all parameterizations in both efficiency and robustness. In a large number
of cases, $\lambda$ and $\mu$ outperform all competitors. These results make sense in light of some additional observations, namely, that (1) relative performance of material parameterizations correlates with nonlinearity, which can be affordably quantified in terms of deviation of the misfit function from a quadratic form, and (2) wavespeed-like parameters perform well for phase-based inversions, but not for waveform- or envelope-based inversions. Since a direct relation between wavespeed perturbations and data residuals exists for phase misfit functions, but not for envelope- or waveform-difference misfit, good performance of Lamé parameters relative to wave speed-like parameters in the latter cases is perhaps not surprising.

The choice of one set of material parameters or another can be regarded as a numerical decision, since it affects the conditioning of an inversion without any change to the underlying misfit function. Because poor numerical conditioning can lead to global convergence difficulties, the material parameterization has a bearing not only on computational efficiency, but also on the ultimate success or failure of an inversion. Such facts were recognized early on in geophysics and other computational fields (e.g., Tarantola 1986, Cai and Keyes 2002).

In one of the earliest and most influential elastic waveform inversion studies, Tarantola (1986) considered the choice of material parameters and recommended compressional- and shear-wave speed over Lamé parameters. This recommendation, it appears, was based mostly on analysis of diffraction pattern responses to different material perturbations; other types of experiments are mentioned, but only in passing. In later work, Tarantola advocated bulk and shear moduli, which display an improved scaling and reduced covariance relative to $\alpha$ and $\beta$.

While the diffraction pattern analysis Tarantola developed has proven useful, it seems unlikely he would have relied so heavily on the device today, given the vastly greater computational resources available. “Brute force” experiments involving hum-
dreds of inversions now provide a feasible, and we argue, much more useful method for comparing material parameterizations.

### 3.1 Theory

The gradient of a full-waveform misfit function in an isotropic elastic inversion can be written as

\[
\delta \chi = \int K_\rho \delta \ln \rho + K_\kappa \delta \ln \kappa + K_\mu \delta \ln \mu \, dx ,
\]

(3.1)

where \( \rho, \kappa, \mu \) are density, bulk modulus and shear modulus, respectively, and \( K_\rho, K_\kappa, K_\mu \) are kernels relating variations in the above material parameters to data misfit. Formulas for the kernels

\[
K_\rho(x) = \int \rho(x) \partial_t s(x, t) \cdot \partial_t s^\dagger(x, t) \, dt 
\]

(3.2)

\[
K_\kappa(x) = \int \kappa(x) \nabla \cdot s(x, t) \nabla \cdot s^\dagger(x, t) \, dt 
\]

(3.3)

\[
K_\mu(x) = \int 2\mu(x) D(x, t) : D^\dagger(x, t) \, dt 
\]

(3.4)

resemble classical imaging conditions, involving interaction between forward and adjoint wavefields \( s \) and \( s^\dagger \) or associated deviatoric strain tensors \( D \) and \( D^\dagger \) (Tromp et al., 2005).

Of course, \( \delta \chi \) can be expressed in other ways as well. For example, kernels for Lamé parameters corresponding to \( \delta \lambda \) and \( \delta \mu \) with density held fixed are given by

\[
K_\lambda = \left(1 - \frac{2\mu}{3\kappa}\right) K_\kappa
\]

(3.5)

\[
K_\mu' = \frac{2\mu}{3\kappa} K_\kappa + K_\mu.
\]

(3.6)
Similarly, kernels for compressional- and shear-wave speed with fixed density are

$$K_\alpha = 2 \left( 1 + \frac{4\mu}{3\kappa} \right) K_\kappa$$  \hspace{1cm} (3.7)

$$K_\beta = 2 \left( K_\mu - \frac{4\mu}{3\kappa} K_\kappa \right),$$  \hspace{1cm} (3.8)

and kernels for bulk-sound speed (i.e., $\phi = \sqrt{\frac{\kappa}{\rho}}$) and shear-wave speed with fixed density are

$$K_{\phi} = 2K_\kappa$$  \hspace{1cm} (3.9)

$$K_{\beta}' = 2K_\mu.$$  \hspace{1cm} (3.10)

More details concerning eqs. 1–8 can be found in Tromp et al. (2005), Zhu et al. (2009) and Luo et al. (2013).

### 3.2 Test problems

To compare material parameterizations, we ran inversions with the following target models: (a) Marmousi onshore, (b) Marmousi offshore, (c) overthrust onshore, (d) overthrust offshore, (e) BP 2007 diapir, (f) BP 2007 anticline. For (a) and (c) we used the standard IFP Marmousi and SEG/EAGE overthrust models, and for (b) and (d) we added a 500 m water layer. For (e) and (f) we cropped the full BP 2007 model between $x = 20–45$ km and $x = 37.5–62.5$ km, respectively, and down-scaled in both $x$ and $z$ by a factor of two. Shear-wave speeds were obtained from compressional-wave speeds by $\beta = \frac{1}{\sqrt{3}} \alpha$, and densities were obtained using Gardner’s law $\rho = 0.31\alpha^{0.25}$. Starting models were derived from target models by convolving with a Gaussian of standard deviation 600 m.
We carried out inversions with waveform-, envelope- and phase-difference misfit functions given by

$$\chi_1(m) = \sum \int |s(m, t) - d(t)|^2 \, dt$$

(3.11)

$$\chi_2(m) = \sum \int |E_s(m, t) - E_d(t)|^2 \, dt$$

(3.12)

$$\chi_3(m) = \sum \int |\phi_s(m, t) - \phi_d(t)|^2 \, dt,$$

(3.13)

where $s$ are synthetics, $d$ are data, $E$ is envelope, $\phi$ is phase, and the sum is taken over all sources, receivers, and components. Rather than instantaneous phase (e.g., Bozdag et al. 2011), we defined $\phi$ as the analytic signal divided by the envelope. This new phased-based misfit function is effective, we find, in reducing cycle-skipping artifacts and probably deserves a more detailed examination like that given to $\chi_2$ by Yuan et al. (2015).

Borrowing a device from the numerical optimization literature, we measured the nonlinearity of each model/misfit combination in terms of the size of the third derivatives of the misfit function (Nash and Nocedal, 1991). Because the third derivatives vanish if the misfit function is quadratic, the expression

$$DQ = \frac{\|g|_{m_{\text{init}}} - g|_{m_{\text{true}}} - H|_{m_{\text{true}}} p\|_{\infty}}{\|p\|_{\infty}^2},$$

(3.14)

where $g$ is the gradient, $H$ is the Hessian and $p = m_{\text{init}} - m_{\text{true}}$, provides a measure of deviation from a quadratic form. We find that the resulting ranking, shown in Table 1, agrees with our subjective sense of how easy or hard each test problem is (with the somewhat odd exception of the BP 2007 anticline test case, which has an unexpectedly high DQ).
3.3 Testing procedures

For each model/misfit/material combination, we carried out a suite of inversions, each with the same starting model but with different frequency bands. While varying the frequency band between inversion, we kept the dominant frequency fixed within each inversion, forgoing any type of multiscale procedure in the manner of Bunks et al. (1995).

In these experiments, forward and adjoint simulations were carried out using SPECFEM2D, with full elastic/acoustic coupling for offshore problems (Komatitsch and Vilotte, 1998a; Luo et al., 2013). Nonlinear optimization, data preprocessing, gradient post-processing, and workflow integration tasks were performed in the SeisFlows framework (http://github.com/PrincetonUniversity/seisflows). In generating data and synthetics, an absorbing boundary condition was used to exclude multiple reflections.

To avoid problematic tradeoffs between parameters, we treated density as a dependent variable through Gardner’s relation. While such relations can be used to modify the gradient directly, i.e., by substituting $\delta \rho = \frac{\partial \rho}{\partial \kappa} \delta \kappa + \frac{\partial \rho}{\partial \mu} \delta \mu$ in eq. 1, a more effective approach, we find, is to simply drop the $K \rho \delta \rho$ term, that is, to update all parameters aside from density by their respective kernels and then, using these new values, update density by its empirical relation.

To avoid complicated choices about penalty function weights, we employed a simple “regularization by convolution” approach in which kernels were convolved with a Gaussian with a fixed standard deviation of 5 numerical grid points, or roughly 100 m (e.g., Tape et al., 2007).

To ensure meaningful comparisons between inversion results, close attention was paid to the nonlinear optimization procedure. Inversions were run using L-BFGS with a memory of five previous gradient evaluations and with M3 scaling (Liu and Nocedal, 1989), stopping after convergence to a minimum of the misfit function or 100 model...
updates, whichever occurred first. For comparison, a subset of inversions was rerun using truncated Newton model updates with Eisenstat-Walker stopping condition and L-BFGS preconditioner, stopping after convergence or 50 model updates. A backtracking line search with bound constraints was used with both types of search directions (Dennis and Schnabel, 1996).

Numerical safeguards are important in L-BFGS inversions because the accuracy of the quasi-Newton approximation to the inverse Hessian is known to break down on occasion. Besides line search bounds, restarting the nonlinear optimization algorithm in the manner of Powell (1977) is essential. By restarting when the angle between the steepest descent direction and search direction exceeds a certain threshold, say 85 degrees, we adapt Powell’s restart condition to L-BFGS. Restarting is usually effective, we find, in restoring fast convergence if an inversion becomes stalled. Details about the number of restarts in the waveform-difference ($\chi_1$) inversions are given in Table 2. Compared with acoustic inversion, the need for restarting in elastic inversion is much greater.

To rate the performance of an inversion, we used the following measure of error reduction

$$\Delta E = \sum_{i=1}^{N} w_i \left( \|m_{i-1} - m_{\text{true}}\| - \|m_i - m_{\text{true}}\| \right),$$  \hspace{1cm} (3.15)

where $N$ is the total number of model updates. Unlike the choice of parameterization to compute model updates with, the choice of one parameterization or another in eq. 15 is not especially significant. To permit comparisons, we converted other parameter sets to $\alpha$ and $\beta$ and used $\|m\| = \|m_\alpha\|_2 + \|m_\beta\|_2$ in calculating $\Delta E$, where $\| \cdot \|_2$ is the $L_2$ norm. Letting $F_i$ be the cumulative number of function and gradient evaluations through the $i$-th model update, our choice of weights

$$w_i = (F_i - F_i-1)^{-1} 2^{-0.01F_i},$$  \hspace{1cm} (3.16)
rewards progress early in an inversion over progress later on. Such weights are impor-
tant because without them eq. 15 would reduce to $|m_0 - m_{true}| - |m_N - m_{true}|$. It follows that some type of weighting is required to distinguish between two inversions that converge to the same result at different computational expense.

### 3.4 Results

Tables 3 and 4 list which parameter sets, of the four considered, performed best and worst in the quasi-Newton inversions in terms of $\Delta E$. Asterisks denote near ties—in which the winner came within 2.5 percent of the next-closest competitor. For illustration, convergence plots for the Marmousi onshore 3 Hz quasi-Newton inversions are shown in Figure 1. Plots for all other test cases are available online (http://github.com/rmodrak/SEG-2016_abstract).

In the results, strong problem dependence is evident with major differences between onshore and offshore test cases related to underlying differences in nonlinearity. Strikingly, $\lambda$ and $\mu$ ranked first in more than 30 percent and $\alpha$ and $\beta$ ranked last in more than 65 percent of test cases—despite the latter’s physical relevance and the former’s lack thereof. Besides poor scaling and large covariance between $\alpha$ and $\beta$, much of this behavior, we believe, can be explained in terms of relationships between material parameters and data misfit—in particular, the fact that wave speed perturbations are more directly related to phase differences than waveform or envelope differences.

To check whether our rankings depended strongly on any aspect of our L-BFGS implementation, we compared quasi-Newton with truncated Newton inversion results. Truncated Newton was less computationally efficient than L-BFGS, which is not inconsistent with Métivier et al. (2014) and not unexpected given the lack of sophisticated preconditioning (Akcelik et al., 2002). Although inversions based on $\alpha$
and $\beta$ did tend to fail less often with truncated Newton model updates, rankings were not significantly affected. Leaving aside computational cost differences, the similarity between L-BFGS and truncated Newton results, in our view, suggests that robustness is more a matter of numerical safeguards than a question of one nonlinear optimization algorithm over another.

### 3.5 Conclusions

Results of “brute force” experiments carried out with attention to numerical details demonstrate that the choice of material parameters in elastic waveform inversion is, at the very least, more complicated than the current literature reflects. Additional tests are needed to explore why, despite theoretical considerations that seem to suggest otherwise, Lamé parameters are an effective choice, and to consider issues such as noisy data, frequency- versus time-domain implementation, and implications for anisotropic problems. Whatever the ultimate findings, it seems clear that, in addition to physical or seismological insights, numerical insights are needed for progress in elastic waveform inversion.
<table>
<thead>
<tr>
<th>Problem</th>
<th>DQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marmousi onshore 3 Hz</td>
<td>$10^5$</td>
</tr>
<tr>
<td>Marmousi onshore 4 Hz</td>
<td>$10^6$</td>
</tr>
<tr>
<td>Marmousi onshore 5 Hz</td>
<td>$10^6$</td>
</tr>
<tr>
<td>Marmousi onshore 6 Hz</td>
<td>$10^6$</td>
</tr>
<tr>
<td>Marmousi onshore 7 Hz</td>
<td>$10^7$</td>
</tr>
<tr>
<td>overthrust onshore 3 Hz</td>
<td>$10^6$</td>
</tr>
<tr>
<td>overthrust onshore 4 Hz</td>
<td>$10^6$</td>
</tr>
<tr>
<td>overthrust onshore 5 Hz</td>
<td>$10^7$</td>
</tr>
<tr>
<td>overthrust onshore 6 Hz</td>
<td>$10^7$</td>
</tr>
<tr>
<td>overthrust onshore 7 Hz</td>
<td>$10^8$</td>
</tr>
<tr>
<td>overthrust offshore 3 Hz</td>
<td>$10^{11}$</td>
</tr>
<tr>
<td>overthrust offshore 4 Hz</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>overthrust offshore 5 Hz</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>overthrust offshore 6 Hz</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>overthrust offshore 7 Hz</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>BP 2007 diapir 3 Hz</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>BP 2007 diapir 4 Hz</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>BP 2007 anticline 3 Hz</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>BP 2007 anticline 4 Hz</td>
<td>$10^{13}$</td>
</tr>
<tr>
<td>BP 2007 anticline 5 Hz</td>
<td>$10^{13}$</td>
</tr>
<tr>
<td>Marmousi offshore 3 Hz</td>
<td>$10^{12}$</td>
</tr>
<tr>
<td>Marmousi offshore 4 Hz</td>
<td>$10^{13}$</td>
</tr>
</tbody>
</table>

Table 3.1: Deviation from quadratic. The above ranking, based on waveform-difference misfit ($\chi_1$) agrees mostly with our subjective sense of difficulty.
Table 3.2: Number of restarts in the waveform-based ($\chi_1$) inversions. A high number of restarts, we find, does not necessarily indicate a given parameterization is performing badly.
Table 3.3: Best-performing material parameterization in terms of model error reduction ($\Delta E$). Asterisk indicates a near tie. Dash denotes all parameterizations failed.
<table>
<thead>
<tr>
<th>Problem</th>
<th>Envelope</th>
<th>Waveform</th>
<th>Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Marmousi onshore 3 Hz</td>
<td>α, β</td>
<td>λ, μ</td>
<td>κ, μ</td>
</tr>
<tr>
<td>Marmousi onshore 4 Hz</td>
<td>α, β</td>
<td>κ, μ*</td>
<td>α, β</td>
</tr>
<tr>
<td>Marmousi onshore 5 Hz</td>
<td>α, β</td>
<td>α, β</td>
<td>—</td>
</tr>
<tr>
<td>Marmousi onshore 6 Hz</td>
<td>α, β*</td>
<td>α, β</td>
<td>—</td>
</tr>
<tr>
<td>Marmousi onshore 7 Hz</td>
<td>κ, μ</td>
<td>α, β</td>
<td>—</td>
</tr>
<tr>
<td>overthrust onshore 3 Hz</td>
<td>α, β</td>
<td>φ, β</td>
<td>α, β</td>
</tr>
<tr>
<td>overthrust onshore 4 Hz</td>
<td>α, β</td>
<td>α, β*</td>
<td>α, β</td>
</tr>
<tr>
<td>overthrust onshore 5 Hz</td>
<td>α, β</td>
<td>α, β</td>
<td>α, β*</td>
</tr>
<tr>
<td>overthrust onshore 6 Hz</td>
<td>α, β</td>
<td>α, β</td>
<td>κ, μ</td>
</tr>
<tr>
<td>overthrust onshore 7 Hz</td>
<td>α, β</td>
<td>λ, μ</td>
<td>κ, μ*</td>
</tr>
<tr>
<td>overthrust offshore 3 Hz</td>
<td>α, β</td>
<td>α, β</td>
<td>α, β</td>
</tr>
<tr>
<td>overthrust offshore 4 Hz</td>
<td>—</td>
<td>κ, μ*</td>
<td>α, β</td>
</tr>
<tr>
<td>overthrust offshore 5 Hz</td>
<td>—</td>
<td>κ, μ</td>
<td>—</td>
</tr>
<tr>
<td>overthrust offshore 6 Hz</td>
<td>—</td>
<td>α, β</td>
<td>—</td>
</tr>
<tr>
<td>overthrust offshore 7 Hz</td>
<td>—</td>
<td>α, β</td>
<td>—</td>
</tr>
<tr>
<td>BP 2007 diapir 3 Hz</td>
<td>—</td>
<td>κ, μ</td>
<td>α, β</td>
</tr>
<tr>
<td>BP 2007 diapir 4 Hz</td>
<td>—</td>
<td>λ, μ</td>
<td>—</td>
</tr>
<tr>
<td>BP 2007 anticline 3 Hz</td>
<td>—</td>
<td>α, β</td>
<td>φ, β</td>
</tr>
<tr>
<td>BP 2007 anticline 4 Hz</td>
<td>—</td>
<td>α, β</td>
<td>α, β</td>
</tr>
<tr>
<td>BP 2007 anticline 5 Hz</td>
<td>—</td>
<td>α, β*</td>
<td>—</td>
</tr>
<tr>
<td>Marmousi offshore 3 Hz</td>
<td>—</td>
<td>φ, β</td>
<td>λ, μ</td>
</tr>
<tr>
<td>Marmousi offshore 4 Hz</td>
<td>—</td>
<td>α, β</td>
<td>—</td>
</tr>
</tbody>
</table>

Table 3.4: Worst-performing material parameterization in terms of model error reduction ($\Delta E$). Asterisk indicates a near tie. Dash denotes all parameterizations failed.
Chapter 4

Anisotropic imaging with fast recovery of tilt and azimuth angles

We present 3D anisotropic imaging results using transverse isotropy parameters together with tilt and azimuth angles. The resulting parametrization can be used to provide full spatial variation in the orientation of the transverse isotropy axis, or by restricting values of the two rotation angles, can be made to reduce to conventional VTI, HTI, or TTI cases. Using the new combination of parameters in an imaging procedure based on the adjoint method, we demonstrate advantages of the new approach for resolving complex anisotropic structures. With a strategy to mitigate nonlinearity, we show that spatially variable tilt and azimuth angles can be quickly recovered from a starting model without such variations. These capabilities are demonstrated through tests with a 3D model of a salt dome.

An anisotropic medium is completely described by an elastic stiffness tensor with 21 independent elements. Studying anisotropy in this general form provides valuable constraints (Rusmanugroho and McMechan, 2010, 2012a,b), but comes with high data acquisition costs.
A more common approach to anisotropy is to make assumptions that reduce the number of independent stiffness tensor elements. In many cases, wave speed variations are more or less symmetric about a single axis so that a transversely isotropic model accounts for dominant anisotropic effects while avoiding the need to deal with other presumably harder-to-constrain anisotropy parameters.

In more complex settings, assumptions about a single symmetry axis break down, and transversely isotropic imaging and inversion procedures experience problems as a result. To provide better imaging in such environments, Duveneck and Bakker (2011) added a tilt angle to the wave speed and Thomsen parameters conventionally used to describe a transversely isotropic medium. By allowing spatial variation in the symmetry axis, their approach helped in resolving complicated, steeply dipping structures.

In the following, we extend Duveneck and Bakker’s method from 2D to 3D, adding an azimuth angle as well as a tilt angle. In the process, we reformulate the imaging procedure and address the fact that Thomsen parameters and rotation angles exacerbate problems with nonlinearity and convergence in iterative imaging and inversion. To avoid such problems, we suggest use of an alternate parametrization (Chen and Tromp, 2007), through which it is possible to recover azimuth or tilt angles in just a few imaging or model update iterations. Recovery of rotation angle parameters can be carried out either in the imaging procedure or in a preliminary velocity model building step, reducing the need for expensive yet otherwise effective migration velocity analysis (Oropeza and McMechan, 2014).
4.1 Theory

In the most general case, the gradient of a waveform misfit function $\chi$ can be expressed as

$$\delta \chi = \int (\delta \rho K_\rho + \delta c :: K_c) \, d^3 x,$$

(4.1)

where $\delta \rho$ and $\delta c$ are variations in mass density and elastic stiffness tensor, and $K_\rho$ and $K_c$ are kernels relating the variation of the misfit function to the above material parameters. Mathematical expressions for the kernels, involving the cross correlation of a downgoing wavefield $s$ with an upgoing wavefield $s^\dagger$, have been derived (Tarantola, 1984; Tromp et al., 2005) and found to resemble classical imaging conditions:

$$K_\rho(x) = - \int \partial_t s^\dagger(x, T-t) \cdot \partial_t s(x, t) \, dt$$

(4.2)

$$K_c(x) = - \int \nabla s^\dagger(x, T-t) \nabla s(x, t) \, dt.$$

(4.3)

For a general anisotropic medium, equation 4.1 can be stated in terms of Voigt parameters $C_{11}, \ldots, C_{66}$. For a vertically transverse isotropic medium, equation 4.1 can be restated more compactly in terms of wave speed and Thomsen parameters $\alpha, \beta, \delta, \epsilon, \gamma$. Despite the sparse representation afforded the stiffness tensor, high non-linearity and slow convergence associated with these five parameters create difficulties for imaging and inversion (Lee et al., 2010).

To extend Duveneck and Bakker’s approach from 2D to 3D, tilt and azimuth parameters $\phi, \theta$ could be added to $\alpha, \beta, \delta, \epsilon, \gamma$ to allow spatial variation in the transverse isotropy axis. Addition of rotation angles to wavespeed and Thomsen parameters, however, further aggravates convergence problems. As a workaround, we suggest use of an alternate parametrization (Chen and Tromp, 2007) that is effective in mitigating nonlinearity. For a rotated transversely isotropic medium, all 21 Chen and Tromp parameters take nonzero values. Only a subspace of rank 7, however,
is required to fully characterize the medium. Contributions that arise outside this subspace can be discarded so that the dimension of the underlying basis remains the same as in the case of 5 transverse isotropy parameters and 2 rotation angles.

With the Chen and Tromp parametrization, equation 4.1 can be rewritten

\[
\delta \chi = \int (\delta \rho K_\rho + \delta A K_A + \delta C K_C + \delta N K_N + \delta L K_L \\
+ \delta F K_F + \delta J_{s,c} K_{J_{s,c}} + \delta K_{s,c} K_{K_{s,c}} + \delta M_{s,c} K_{M_{s,c}} \\
+ \delta G_{s,c} K_{G_{s,c}} + \delta B_{s,c} K_{B_{s,c}} + \delta H_{s,c} K_{H_{s,c}} \\
+ \delta D_{s,c} K_{D_{s,c}} + \delta E_{s,c} K_{E_{s,c}}) \, d^3x. 
\] (4.4)

For a rotated transversely isotropic medium with angles \( \phi \) and \( \theta \), relations between Chen and Tromp parameters and Love parameters (a subset of Voigt parameters) are found to be

\[
A = C_{11} - \frac{1}{64} \left[ (7\lambda_1 + 16\lambda_2) - 4(\lambda_1 + 4\lambda_2) \cos 2\theta \\
- 3\lambda_1 \cos 4\theta \right], 
\] (4.5)

\[
C = C_{33} + \frac{1}{8} \left[ (4\lambda_2 - \lambda_1) - 4\lambda_2 \cos 2\theta + \lambda_1 \cos 4\theta \right] \\
= C_{11} - \frac{1}{8} \left[ (\lambda_1 + 4\lambda_2) + 4\lambda_2 \cos 2\theta - \lambda_1 \cos 4\theta \right], 
\] (4.6)

\[
L = C_{44} + \frac{1}{16} \left[ (\lambda_1 + 4\lambda_3) - 4\lambda_3 \cos 2\theta - \lambda_1 \cos 4\theta \right], 
\] (4.7)

\[
N = C_{66} + \frac{1}{64} \left[ (3\lambda_1 - 32\lambda_3) \\
- 4(\lambda_1 - 8\lambda_3) \cos 2\theta + \lambda_1 \cos 4\theta \right] \\
= C_{44} + \frac{1}{64} \left[ (3\lambda_1 + 32\lambda_3) \\
- 4(\lambda_1 - 8\lambda_3) \cos 2\theta + \lambda_1 \cos 4\theta \right], 
\] (4.8)

69
\[ F = C_{13} + \frac{1}{16} [(3\lambda_1 - 8\lambda_3 + 2\lambda_2) \]
\[ - (2\lambda_1 - 8\lambda_3 + 2\lambda_2) \cos 2\theta - \lambda_1 \cos 4\theta] \]
\[ = C_{11} - 2C_{44} - \frac{1}{16} [(5\lambda_1 + 8\lambda_3 + 6\lambda_2) \]
\[ + (2\lambda_1 - 8\lambda_3 + 2\lambda_2) \cos 2\theta + \lambda_1 \cos 4\theta] , \]

\[ J_s = \frac{1}{64} [2(\lambda_1 + 4\lambda_2) \sin 2\theta + 3\lambda_1 \sin 4\theta] \sin \phi, \]

\[ J_c = \frac{1}{64} [2(\lambda_1 + 4\lambda_2) \sin 2\theta + 3\lambda_1 \sin 4\theta] \cos \phi, \]

\[ K_s = \frac{1}{64} \lambda_1 (4 \sin 2\theta + 7 \sin 4\theta) \sin \phi, \]

\[ K_c = \frac{1}{64} \lambda_1 (4 \sin 2\theta + 7 \sin 4\theta) \cos \phi, \]

\[ M_s = \frac{1}{32} [2(\lambda_1 - 8\lambda_3) \sin 2\theta - \lambda_1 \sin 4\theta] \sin \phi, \]

\[ M_c = - \frac{1}{32} [2(\lambda_1 - 8\lambda_3) \sin 2\theta - \lambda_1 \sin 4\theta] \cos \phi, \]

\[ G_s = - \frac{1}{16} [(\lambda_1 - 4\lambda_3) + 4\lambda_3 \cos 2\theta - \lambda_1 \cos 4\theta] \sin 2\phi, \]

\[ G_c = \frac{1}{16} [(\lambda_1 - 4\lambda_3) + 4\lambda_3 \cos 2\theta - \lambda_1 \cos 4\theta] \cos 2\phi, \]

\[ B_s = \frac{1}{16} [(\lambda_1 + 4\lambda_2) - 4\lambda_2 \cos 2\theta - \lambda_1 \cos 4\theta] \sin 2\phi, \]

\[ B_c = - \frac{1}{16} [(\lambda_1 + 4\lambda_2) - 4\lambda_2 \cos 2\theta - \lambda_1 \cos 4\theta] \cos 2\phi, \]

\[ H_s = \frac{1}{16} [(3\lambda_1 - 8\lambda_3 + 2\lambda_2) \]
\[ - (2\lambda_1 - 8\lambda_3 + 2\lambda_2) \cos 2\theta - \lambda_1 \cos 4\theta] \sin 2\phi, \]

\[ H_c = - \frac{1}{16} [(3\lambda_1 - 8\lambda_3 + 2\lambda_2) \]
\[ - (2\lambda_1 - 8\lambda_3 + 2\lambda_2) \cos 2\theta - \lambda_1 \cos 4\theta] \cos 2\phi, \]

\[ D_s = - \frac{1}{32} \lambda_1 (2 \sin 2\theta - \sin 4\theta) \sin 3\phi, \]

\[ D_c = - \frac{1}{32} \lambda_1 (2 \sin 2\theta - \sin 4\theta) \cos 3\phi, \]
\[ E_s = -\frac{1}{64} \lambda_1 (3 - 4 \cos 2\theta + \cos 4\theta) \sin 4\phi, \quad (4.24) \]
\[ E_c = \frac{1}{64} \lambda_1 (3 - 4 \cos 2\theta + \cos 4\theta) \cos 4\phi, \quad (4.25) \]

with
\[ \lambda_1 = C_{11} + C_{33} - 2C_{13} - 4C_{44}, \quad (4.26) \]
\[ \lambda_2 = C_{11} - C_{33}, \quad (4.27) \]
\[ \lambda_3 = C_{66} - C_{44}. \quad (4.28) \]

Corresponding relations between kernels are found to be
\[ K_A = K_{C_{11}} + K_{C_{12}} + K_{C_{22}}, \quad (4.29) \]
\[ K_C = K_{C_{33}}, \quad (4.30) \]
\[ K_N = K_{C_{66}} - 2K_{C_{12}}, \quad (4.31) \]
\[ K_L = K_{C_{44}} + K_{C_{55}}, \quad (4.32) \]
\[ K_F = K_{C_{13}} + K_{C_{23}}, \quad (4.33) \]
\[ K_J_c = \frac{8}{3} (K_{C_{15}} + K_{C_{25}} + K_{C_{35}}), \quad (4.34) \]
\[ K_J_s = 2(K_{C_{14}} + K_{C_{24}} + K_{C_{34}}), \quad (4.35) \]
\[ K_{K_s} = -2K_{C_{34}}, \quad (4.36) \]
\[ K_{K_c} = -2K_{C_{35}}, \quad (4.37) \]
\[ K_{M_s} = K_{C_{14}} + K_{C_{24}} - 2K_{C_{56}}, \quad (4.38) \]
\[ K_{M_c} = \frac{1}{5} (3K_{C_{15}} - 7K_{C_{25}} + 3K_{C_{35}}) + K_{C_{46}}, \quad (4.39) \]
\[ K_{G_s} = -K_{C_{45}}, \quad (4.40) \]
\[ K_{Gc} = -(K_{C44} - K_{C55}), \quad (4.41) \]
\[ K_{Bs} = -\frac{1}{2}(K_{C16} + K_{C26}), \quad (4.42) \]
\[ K_{Bc} = K_{C11} - K_{C22}, \quad (4.43) \]
\[ K_{Hs} = -K_{C36}, \quad (4.44) \]
\[ K_{Hc} = K_{C13} - K_{C23}, \quad (4.45) \]
\[ K_{Ds} = 2(K_{C14} - K_{C24}), \quad (4.46) \]
\[ K_{Dc} = -(K_{C25} + K_{C36}), \quad (4.47) \]
\[ K_{Es} = -(K_{C16} - K_{C26}), \quad (4.48) \]
\[ K_{Ec} = K_{C11} - K_{C12} + K_{C22} - K_{C66}. \quad (4.49) \]

In principle, any of the kernels (4.29–4.49) can be used for imaging. In practice, we use a particular combination known as the impedance kernel,

\[ K'_\rho = \rho K_\rho + c : \mathbf{K}_e, \quad (4.50) \]

which provides an especially crisp result (Luo et al., 2013).

### 4.2 Imaging experiments

The 3D anisotropic elastic model used to test the imaging procedure is shown in Figure 1. Following Orpeza and McMechan, we start with a subregion of the 2007 BP model. We use this 2D subregion to create a 3D rotationally symmetric model of a salt dome, with \( P \)-wave speed taken from the BP model, \( S \)-wave speed obtained by scaling \( P \)-wave speed by a factor 1.8, density obtained by Gardner’s equation \( \rho = 0.31\alpha^{0.25} \), azimuth angles chosen to be radially symmetric with respect to the
center of the salt dome, and tilt angles as well as Thomsen parameters taken from the BP model. The size of the resulting model is $8 \times 8 \times 3.3$ km, including the 0.2 km water layer.

For the imaging experiments, data were generated from the salt dome model using a spectral element method, avoiding stability problems that have been reported with some anisotropic finite difference methods. Seismic traces from $\sim 100$ sources were simulated at $\sim 30000$ receivers. Impedence kernels were computed by backprojecting differences between data generated from the 3D model and synthetics generated from a slightly smoothed version of the same model. Comparisons with conventional VTI and HTI imaging methods were made by restricting the spatial variation of tilt and azimuth in the smoothed models, but otherwise using the same background velocity model.

Before showing results of these comparisons, we first illustrate the effectiveness of the impedance kernel as a tool for imaging. Results from two conventional imaging experiments, using isotropic and VTI imaging operators on VTI data, are shown in Figure 2. In both cases, impedance kernels are free from diving wave artifacts, avoiding the need for Laplacian filtering or other postprocessing used in conventional migration.

Returning to use of the new imaging procedure, Figure 3 provides a comparison of the new approach with conventional VTI and HTI approaches. In the VTI imaging result, vertical resolution is limited. The salt dome is nicely located and focused, but nearly vertical reflectors on the flanks are not well recovered. In the HTI imaging result, horizontal resolution is limited. Flat reflectors in the lower part of the model are not well imaged. One reflector near 1.0 km is missing, and reflectors below 2.0 km show up much deeper than expected.

In comparison to conventional VTI and HTI methods, the new imaging method provides improved horizontal and vertical resolution, producing a focused image in
which all reflectors are correctly located. In considering this result, it is important to keep in mind that the improved performance of the new imaging procedure rests on the availability of azimuth and tilt angles for the background velocity model. In a final experiment, we demonstrate that these rotation angles can be quickly and accurately recovered.

To recover azimuth, we note from equations (4.5–4.25) that ratios \( \frac{J_s}{J_c}, \frac{K_s}{K_c}, \frac{M_s}{M_c}, \frac{G_s}{G_c}, \frac{B_s}{B_c}, \frac{H_s}{H_c}, \frac{D_s}{D_c}, \frac{E_s}{E_c} \) depend solely on \( \phi \). By updating an isotropic or VTI starting model via the gradient given in equation (4.4), the azimuth, or in other words the projection of the anisotropic fast axis onto the azimuthal plane, can be obtained as \( \phi = \frac{1}{2} \arctan(\frac{G_s}{G_c}) \). Comparison between true azimuth and recovered azimuth after a single model update from a VTI starting model is shown in Figure 4.

**Acknowledgements**

We thank ConocoPhillips for supporting this research.
Figure 4.1: 3D model of a salt dome. (a) Thomsen and wavespeed parameters $\rho$, $\alpha$, $\beta$, $\epsilon$, $\delta$, $\gamma$ are shown in vertical cross section through the center of the salt dome. (b) Tilt $\theta$ and azimuth $\phi$ are shown in vertical and horizontal cross section.
Figure 4.2: Conventional VTI imaging result demonstrating the effectiveness of the impedance kernel as a tool for imaging. The impedance kernel, shown above without the use of any filtering or other postprocessing steps, was computed using a VTI forward modeling operator and VTI background velocity model. Data were generated from the 3D salt model with spatial variation of tilts and azimuths restricted to reduce to the VTI case.

Figure 4.3: Comparison of the new imaging procedure with conventional HTI and VTI approaches. HTI and VTI forward modeling operators and background velocity models were used for the left and center panels, respectively, while an imaging procedure based on the new rotated transverse isotropy parametrization was used for the right panel. In all cases, data were generated from the 3D salt model with full spatial variation in tilts and azimuths. Pronounced artifacts, present in the HTI and VTI cases, are absent from the new imaging result.
Figure 4.4: Comparison between true azimuth and recovered azimuth after a single model update from a VTI starting model. (a) True azimuth. (b) Recovered azimuth.
Chapter 5

Anisotropic inversion

By allowing spatial variations in the direction of the anisotropic fast axis, tilted transverse isotropy (TTI) helps to image complex or steeply dipping structures. Without a priori geologic constraints, however, recovery of all the anisotropic parameters can be nontrivial and nonunique. We propose two methods for TTI inversion with tilt-angle recovery, one based on the familiar Voigt parameters, and another based on the so-called Chen & Tromp parameters known from regional and global seismology. Both parameterizations arise naturally in seismic wave propagation and facilitate straightforward recovery of the tilt angle and anisotropic strength. In numerical experiments with vertical transversely isotropic (VTI) starting models and TTI target models, we show that both the Voigt and Chen & Tromp parameters allow quick and robust recovery of steeply dipping anticlinal structures.

5.1 Introduction

The 21 independent elements of the elastic stiffness tensor provide a complete description of the elastic properties of an anisotropic medium. Various fully anisotropic modeling techniques have been proposed, including high-frequency asymptotic methods of the type described by Gajewski and Pšenčík (1990), finite-difference
velocity-stress methods as developed by Tessmer (1995), and spectral-element
displacement-stress methods as developed by Komatitsch and Vilotte, 1998a; Komatitsch and Tromp, 1999; Komatitsch et al., 2000b,a). Such techniques have
been used to characterize cracked reservoirs by simulating multicomponent seismic
amplitude variations with offset and azimuth (AVOAZ) (e.g., Hu and McMechan,
2010; Rusmanugroho and McMechan, 2010). The 21 independent elastic stiffness
can be estimated through the Christoffel equation using phase slowness and polarization vectors measured from vertical seismic profile data (e.g.,
Dewangan and Grecha, 2003; Rusmanugroho and McMechan, 2012a,b).

Chen and Tromp (2007) introduce new elastic parameters for a general anisotropic
medium that arise naturally in seismic wave propagation and, unlike the conventional
Voigt parameters, are independent of any coordinate system. Thirteen of the new
parameters, $A$, $C$, $F$, $L$, $N$, $B_{c,s}$, $H_{s,c}$, $G_{s,c}$, and $E_{s,c}$, deal primarily with surface-wave propagation, and the remaining eight, $J_{s,c}$, $K_{s,c}$, $M_{s,c}$, and $D_{s,c}$, deal primarily with body-wave propagation. An even Fourier series of degrees zero \( \{A, C, F, L, N\} \),
two \( \{B_{c,s}, H_{s,c}, G_{s,c}\} \), and four \( \{E_{c,s}\} \) determines the phase speeds of surface waves, and an odd Fourier series of degree one \( \{J_{s,c}, K_{s,c}, M_{s,c}\} \) and three \( \{D_{c,s}\} \) determines those of body waves. The five parameters $A$, $C$, $L$, $N$, and $F$ suffice to characterize transverse isotropy with a vertical symmetry axis and the remaining 16 parameters correspond to anisotropy which is a function of tilt angle and azimuth.

3D elastic VTI media are commonly described by the vertical wavespeed $\alpha_0$ and
shear wavespeed $\beta_0$ and three unitless anisotropy parameters, $\varepsilon$, $\delta$, and $\gamma$, which are commonly associated with horizontal thin layering and shale (Thomsen, 1986). For 2D acoustic VTI, Alkhalifah and Tsvankin (1995) note that the anisotropic parameters $\alpha_0$, $\varepsilon$, and $\delta$ cannot be accurately resolved by inverting NMO wavespeeds unless one of them is known. For weak anisotropy, they show that the difference $\eta = \varepsilon - \delta$ can be accurately estimated without precisely knowing $\alpha_0$. They also propose an al-
ternative inversion procedure for NMO wavespeeds parameterized in terms of $\alpha_{NMO}$, the zero-dip NMO wavespeed, and $\eta$. These parameters are used to describe nonhyperbolic moveout and the time-migration impulse response.

Tilted transverse isotropy (TTI) is a form of rotated transverse isotropy that can be described by VTI parameters together with two spatially variable rotation angles, namely, azimuth $\phi$ and tilt angle $\theta$. For 2D TTI, the orientation of the anisotropic symmetry axis can be described using only a tilt angle. Grechka and Tsvankin (2000) recover $\alpha_0$, $\varepsilon$, $\delta$, $\phi$, and $\theta$ using $P$-wave NMO ellipses from horizontal and dipping reflectors and show that the uncertainty in $\theta$ propagates into $\varepsilon$ and $\delta$.

The anisotropic symmetry axis is commonly chosen to be orthogonal to the bedding plane of geological structures. Wang and Tsvankin (2013) show that the anisotropic parameters $\alpha_0$, $\varepsilon$, and $\delta$ are recoverable through the moveout of a common-image-gather (CIG) of Kirchhoff migration, which is iterated using ray-based tomography. The symmetry axis is assumed to be perpendicular to reflectors in the stacked migrated section. Oropeza and McMechan (2014) estimate the 2D acoustic TTI parameters $\alpha_0$, $\varepsilon$, and $\delta$ by maximizing the stacked amplitude along the traveltime curve in a common-reflection-point (CRP) gather. During the inversion, $\alpha_0$ is weighted less than $\delta$ and $\varepsilon$ to reduce the dominance of $\alpha_0$. The orientation of the symmetry axis orthogonal to the local reflector orientation is calculated using parsimonious migration (Hua and McMechan, 2003), and errors in $\theta$ are shown to affect the recovery of $\varepsilon$ and $\delta$.

Full-waveform inversion (FWI), an iterative nonlinear optimization procedure, can be used to obtain high-resolution seismic images. Tarantola (1984) showed that the Fréchet derivatives of a waveform-difference misfit function can be computed via the interaction between a forward-propagating wavefield and a reverse-propagating data residual wavefield. For 2D elastic VTI media, Lee et al. (2010) propose a strategy in which the Voigt parameters $C_{13}$, $C_{33}$, $C_{44}$, and $\varepsilon$ are independently updated, and $C_{11}$
is updated using the estimated $C_{33}$ and $\varepsilon$. For 2D acoustic VTI media, Gholami et al. (2013a) consider three different parameterizations, namely, $\{\alpha_0, \delta, \varepsilon\}$, $\{\alpha_{NMO}, \delta, \varepsilon\}$, and $\{\alpha_{NMO}, \delta, \eta\}$. In each of the above parameterizations, the dimensionless parameters are less influential than the wavespeed $\alpha_0$ or $\alpha_{NMO}$; at the final iteration, the inverted Thomsen parameters are similar to the initial ones. These experiments imply that keeping the true Thomsen parameters fixed—for example, obtained from well logs—during the inversion does not significantly affect the recovery of the wavespeeds. Gholami et al. (2013b) note that $C_{11}$ and $C_{33}$ significantly affect the data, and are three times more sensitive than $C_{13}$, which is subject to large trade-off artifacts.

Here, we propose two new approaches for 2D anisotropic FWI based on the Voigt and Chen & Tromp parameterizations. After describing the theory behind the two methods, we compare their performance in numerical experiments with the 2D BP TTI model (http://www.freeusp.org/2007_BP_Ani_Vel_Benchmark/listing.html).

### Waveform inversion for 2D TTI media

For a model $m$ and observed and synthetic data $s^{\text{obs}}$ and $s^{\text{syn}}$ recorded at geophones $x_i$, $i = 1, ..., N$, the least-squares waveform misfit is

$$\chi(m) = \frac{1}{2} \sum_{i=1}^{N} \int |s^{\text{syn}}(x_i, t; m) - s^{\text{obs}}(x_i, t)|^2 dt. \quad (5.1)$$

For a general anisotropic medium, the variation of the misfit function can be expressed as

$$\delta \chi = \int (\delta \rho K_\rho + \delta c : : K_c) d^3 x, \quad (5.2)$$

where $\delta \rho$ and $\delta c$ denote variations in mass density and the fourth-order elastic stiffness tensor, $K_\rho$ and $K_c$ are the corresponding Fréchet derivatives, and $::$ denotes the
double-double dot-product of two fourth-order tensors. The Fréchet derivatives

\[ K_\rho(x) = -\int \partial_t s^\dagger(x, T - t) \cdot \partial_t s(x, t) \, dt, \]  

(5.3)

\[ K_c(x) = -\int \nabla s^\dagger(x, T - t) \nabla s(x, t) \, dt, \]  

(5.4)

are obtained by cross-correlating the forward wavefield \( s \) with the data residual wavefield (e.g., Tarantola, 1984; Tromp et al., 2005)

\[ s^\dagger(x, t) = \sum_{i=1}^{N} \omega_i \left[ s_{i}^{\text{syn}}(T - t) - s_{i}^{\text{obs}}(T - t) \right] \, dt \delta(x - x_i), \]  

(5.5)

where \( \omega_i \) is a weighting function (e.g., Chattopadhyay and McMechan, 2008; Zhu et al., 2009), namely

\[ \omega_i^{-1} = \int |s_i^{\text{obs}}(t)|^2 \, dt. \]  

(5.6)

In practice, it is preferable to work with density-normalized elastic moduli (e.g., Sieminski et al., 2007a,b), that is, \( c' = c/\rho \). In this parameterization we have

\[ \delta \chi = \int \left( \delta \ln \rho K'_\rho + \delta c' :: K'_c \right) \, d^3x, \]  

(5.7)

where

\[ K'_\rho = \rho K_\rho + c :: K_c, \]  

(5.8)

\[ K'_c = \rho K_c. \]  

(5.9)

The kernel \( K'_\rho \) is the impedance kernel. Working in the isotropic elastic case, Zhu et al. (2009) and Luo et al. (2013) show that \( K_\rho \) in equation (5.3) is equivalent to Claerbout’s imaging principle. Douma et al. (2010) show that \( K'_\rho \) resembles reverse-time migration with low-frequency artifact filtering, thereby providing a
natural and effective imaging condition. Rusmanugroho and Tromp (2014) and Rusmanugroho et al. (2015) extend the analysis to transversely isotropic media.

Seismic wave propagation in a 2D VTI medium depends on four Voigt parameters, \( C_{11}, C_{13}, C_{33}, \) and \( C_{55}, \) which may be expressed in terms of vertical \( P- \) and \( S- \) wavespeeds, \( \alpha_0 \) and \( \beta_0, \) and Thomsen parameters, \( \epsilon \) and \( \delta, \) as

\[
C_{11} = \rho \alpha_0^2 (1 + 2\epsilon),
\]

\[
C_{13} = \rho \left[ \alpha_0^2 (1 + 2\delta) - 2\beta_0^2 \right],
\]

\[
C_{33} = \rho \alpha_0^2,
\]

\[
C_{55} = \rho \beta_0^2.
\]

Following (e.g., MacBeth, 2002; Farra et al., 2016), for small \( \delta \) in the range from \(-0.1\) to \(0.2,\)

\[
\delta = \frac{(C_{13} + C_{55})^2 - (C_{33} - C_{55})^2}{2C_{33}(C_{33} - C_{55})}
\]

may be approximated as

\[
\delta = \frac{C_{13} + 2C_{55} - C_{33}}{2C_{33}}.
\]

In 3D VTI, there is axisymmetry about the \( z \) axis. Following Dellinger and Etgen (1990) and Komatitsch et al. (2000a), we have used \( C_{55} = c_{1313} \) instead of \( C_{44} = c_{2323}.\)

Following MacBeth (2002), the relationship between TTI and VTI stiffness matrices, \( C^R \) and \( C, \) may be expressed as

\[
C^R = M_\theta CM_\theta^T,
\]
where

\[
M_\theta = \begin{pmatrix}
\cos^2 \theta & 0 & \sin^2 \theta & 0 & -\sin 2\theta & 0 \\
0 & 1 & 0 & 0 & 0 & 0 \\
\sin^2 \theta & 0 & \cos^2 \theta & 0 & \sin 2\theta & 0 \\
0 & 0 & 0 & \cos \theta & 0 & \sin \theta \\
\frac{1}{2} \sin 2\theta & 0 & -\frac{1}{2} \sin 2\theta & 0 & \cos 2\theta & 0 \\
0 & 0 & 0 & -\sin \theta & 0 & \cos \theta \\
\end{pmatrix}
\]

(5.17)

\[
C = \begin{pmatrix}
C_{11} & C_{11} - 2C_{66} & C_{13} & 0 & 0 & 0 \\
C_{11} & C_{13} & 0 & 0 & 0 \\
C_{33} & 0 & 0 & 0 \\
1998 & C_{55} & 0 \\
C_{55} & 0 \\
\end{pmatrix}
\]

(5.18)

and hence

\[
C^R = \begin{pmatrix}
C_{11}^R & C_{12}^R & C_{13}^R & 0 & C_{15}^R & 0 \\
C_{22}^R & C_{23}^R & 0 & C_{25}^R & 0 \\
C_{33}^R & 0 & C_{35}^R & 0 \\
C_{44}^R & 0 & 0 \\
C_{55}^R & 0 \\
C_{66}^R \\
\end{pmatrix}
\]

(5.19)

The rotation angle \( \theta \) is clockwise about the \( y \) axis. Since we are stating the 2D theory, only a tilt angle rotation is involved.

If a 2D elastic TTI medium (\( y = 0 \)) is considered, the symmetric stiffness (Voigt) matrix is reduced to

\[
C^R = \begin{pmatrix}
C_{11}^R & C_{13}^R & C_{15}^R \\
C_{33}^R & C_{35}^R \\
C_{55}^R \\
\end{pmatrix}
\]

(5.20)
where

\[ C_{11}^R = \frac{1}{8} \left( 3C_{11} + 2C_{13} + 3C_{33} + 4C_{55} + 4\lambda_2 \cos 2\theta + \lambda_1 \cos 4\theta \right), \quad (5.21) \]

\[ C_{13}^R = \frac{1}{8} \left( C_{11} + 6C_{13} + C_{33} - 4C_{55} - \lambda_1 \cos 4\theta \right), \quad (5.22) \]

\[ C_{15}^R = \frac{1}{4} (\lambda_2 \sin 2\theta + \lambda_1 \sin 4\theta), \quad (5.23) \]

\[ C_{33}^R = \frac{1}{8} \left( 3C_{11} + 2C_{13} + 3C_{33} + 4C_{55} - 4\lambda_2 \cos 2\theta + \lambda_1 \cos 4\theta \right), \quad (5.24) \]

\[ C_{35}^R = \frac{1}{4} (\lambda_2 \sin 2\theta - \lambda_1 \sin 4\theta), \quad (5.25) \]

\[ C_{55}^R = \frac{1}{8} \left( C_{11} - 2C_{13} + C_{33} + 4C_{55} - \lambda_1 \cos 4\theta \right), \quad (5.26) \]

with

\[ \lambda_1 = C_{11} - 2C_{13} + C_{33} - 4C_{55}, \quad (5.27) \]

\[ \lambda_2 = C_{11} - C_{33}. \quad (5.28) \]

Noting that the right-hand sides of equations (5.21)–(5.26) involve only one rotation angle \( \theta \) and four unrotated Voigt parameters, a total of five parameters are independent in a 2D elastic TTI medium.

Alternatively, a 2D anisotropic medium may be characterized in terms of the six Chen & Tromp parameters (Sieminski et al., 2007a,b)

\[ A = \frac{3}{8} C_{11}^R, \quad (5.29) \]

\[ C = C_{33}^R, \quad (5.30) \]

\[ L = \frac{1}{2} C_{55}^R, \quad (5.31) \]

\[ F = \frac{1}{2} C_{13}^R. \quad (5.32) \]
\[ J = \frac{3}{8} C_{15}^R, \]  
(5.33)

\[ K = \frac{3}{8} C_{15}^R - \frac{1}{2} C_{35}^R. \]  
(5.34)

For 2D TTI, variation of the misfit function given by equation (5.7) may be written in the alternative forms

\[ \delta \chi = \int \left( \delta \ln \rho K' \rho + \delta C'_{11} K_{C'_{11}} + \delta C'_{13} K_{C'_{13}} + \delta C'_{15} K_{C'_{15}} + \delta C'_{33} K_{C'_{33}} + \delta C'_{35} K_{C'_{35}} + \delta C'_{55} K_{C'_{55}} \right) d^3x, \]  
(5.35)

for a Voigt parameterization, or

\[ \delta \chi = \int \left( \delta \ln \rho K' \rho + \delta A' K_A + \delta C' K_C + \delta L' K_L + \delta F' K_F + \delta J' K_J + \delta K' K_K \right) d^3x, \]  
(5.36)

for a Chen & Tromp parameterization. The two sets of Fréchet derivatives are related via

\[ K_{A'} = \frac{8}{3} K_{C'_{11}}, \]  
(5.37)

\[ K_{C'} = K_{C'_{33}}, \]  
(5.38)

\[ K_{L'} = K_{C'_{55}}, \]  
(5.39)

\[ K_{F'} = 2 K_{C'_{13}}, \]  
(5.40)

\[ K_{J'} = \frac{8}{3} K_{C'_{15}} + 2 K_{C'_{35}}, \]  
(5.41)

\[ K_{K'} = - K_{C'_{35}}. \]  
(5.42)

Because the relationship between anisotropic rotation angles and seismic observables is highly nonlinear, it is best to perform TTI inversions in a parameterization that does not explicitly include the tilt angle. Use of the Voigt or Chen & Tromp pa-
rameters appears effective. Adding the density-normalized equations (5.23) and (5.25) yields

\[ C'_{15} + C'_{35} = \frac{1}{2} (C'_{11} - C'_{33}) \sin 2\theta, \]  

(5.43)

and subtracting density-normalized equation (5.21) from equation (5.24) yields

\[ C'_{11} - C'_{33} = (C'_{11} - C'_{33}) \cos 2\theta. \]  

(5.44)

The tilt angle, calculated from equations (5.43) and (5.44), may be obtained after the fact from an updated model using

\[ \theta = \frac{1}{2} \arctan \frac{2 \left( C'_{15} + C'_{35} \right)}{C'_{11} - C'_{33}}, \]  

(5.45)

or

\[ \theta = \frac{1}{2} \arctan \frac{2 \left( \frac{7}{3} J' + K' \right)}{\frac{8}{3} A' - C'}. \]  

(5.46)

Following Chen and Tromp (2007), a useful measure of the strength of the anisotropy along the tilt angle direction, which is obtained by taking the square root of the sum of the squared numerator and denominator of equation (5.45), is given by

\[ M = \sqrt{4 \left( C'_{15} + C'_{35} \right)^2 + \left( C'_{11} - C'_{33} \right)^2}, \]  

(5.47)

or

\[ M = \sqrt{4 \left( \frac{7}{3} J' + K' \right)^2 + \left( \frac{8}{3} A' - C' \right)^2}. \]  

(5.48)

The fast axis vector is calculated from either equations (5.45) and (5.47), or equations (5.46) and (5.48).

Recalling that 2D elastic TTI models are completely characterized by a five-dimensional subset of four stiffness tensor elements and one rotation angle, equations (5.21)–(5.26) could be used to project from the full stiffness tensor onto a TTI
stiffness tensor. In synthetic inversions with TTI target models, however, it appears that whether or not one projects onto a TTI basis following each model update has little effect on the tilt angle recovered through equations (5.45) or (5.46).

Suppose the TTI Voigt parameters and tilt angle have been recovered by FWI. Then equations (5.21)–(5.26) can be expressed as

\[
\begin{pmatrix}
8C_{11}^{R} \\
8C_{13}^{R} \\
4C_{15}^{R} \\
8C_{33}^{R} \\
4C_{35}^{R} \\
8C_{55}^{R}
\end{pmatrix} = \begin{pmatrix}
3 + 4 \cos 2\theta + \cos 4\theta & 2 - 2 \cos 4\theta & 3 - 4 \cos 2\theta + \cos 4\theta & 4 - 4 \cos 4\theta \\
1 - \cos 4\theta & 6 + 2 \cos 4\theta & 1 - \cos 4\theta & -4 + \cos 4\theta \\
\sin 2\theta + \sin 4\theta & -2 \sin 4\theta & -\sin 2\theta + \sin 4\theta & -4 \sin 4\theta \\
3 - 4 \cos 2\theta + \cos 4\theta & 2 - 2 \cos 4\theta & 3 + 4 \cos 2\theta + \cos 4\theta & 4 - 4 \cos 4\theta \\
\sin 2\theta - \sin 4\theta & 2 \sin 4\theta & -\sin 2\theta - \sin 4\theta & 4 \sin 4\theta \\
1 - \cos 4\theta & -2 + 2 \cos 4\theta & 1 - \cos 4\theta & 4 + 4 \cos 4\theta
\end{pmatrix}
\begin{pmatrix}
C'_{11} \\
C'_{13} \\
C'_{33} \\
C'_{35} \\
C'_{55}
\end{pmatrix}.
\]

The density-normalized Voigt parameters of 2D VTI, \(C'_{11}, C'_{13}, C'_{33},\) and \(C'_{55}\), are retrieved by solving equation (5.49). The vertical wavespeeds, \(\alpha_0\) and \(\beta_0\), and the Thomsen parameters, \(\varepsilon\) and \(\delta\), can be estimated from the VTI Voigt parameters via the inverse relationships (5.10)–(5.13).

### 5.2 Sensitivity analysis

Scattering patterns of a virtual source of partial-derivative or perturbed wavefields are frequently used to measure sensitivity of model parameters as a function of incidence and scattering angles to recorded data. This sensitivity analysis provides knowledge of what types of data are required and which model parameters are well resolved during the inversion. Tarantola (1986) shows analytical radiation patterns of isotropic diffractions generated by incoming \(P\)-, \(SV\)-, and \(SH\)-waves for three model parameterizations, \(\{\lambda, \mu, \rho\}\), \(\{\rho, \ IP, IS\}\), and \(\{\rho, \alpha_0, \beta_0\}\). For example, all three parameters in \(\{\lambda, \mu, \rho\}\), \(IP\) and \(IS\) in \(\{\rho, \ IP, IS\}\), and all three parameters in \(\{\rho, \alpha_0, \beta_0\}\).
\(\alpha_0, \beta_0\) can be retrieved from moderate-offset \(P-P\) data. In VTI acoustic media, Operto et al. (2013) show radiation patterns for two model parameterizations, \(\{\alpha_0, \delta, \varepsilon\}\) (showing that \(\alpha_0\) and \(\varepsilon\) can be recovered), and \(\{\alpha_0, \delta, \alpha_h\}\) (showing that \(\alpha_0\) and \(\alpha_h\) can be constrained).

Following Alkhalifah and Plessix (2014) and Pan et al. (2016), analytical radiation patterns of \(P-P\) and \(P-S\) scattered-waves parameterized in terms of density-normalized Voigt for 2D TTI may be expressed as

\[
\mathbb{R}_{P-P}(\vartheta_1, \vartheta_2, \theta, \Delta C'^R_{11}) = \left( \sin^2 \vartheta_1 \cos^2 \theta + \cos^2 \vartheta_1 \sin^2 \theta - \frac{1}{2} \sin 2\vartheta_1 \sin 2\theta \right) \sin^2 (\vartheta_2 + \theta),
\]

(5.50)

\[
\mathbb{R}_{P-SV}(\vartheta_1, \vartheta_2, \theta, \Delta C'^R_{11}) = \frac{1}{2} \left( \sin^2 \vartheta_1 \cos^2 \theta + \cos^2 \vartheta_1 \sin^2 \theta - \frac{1}{2} \sin 2\vartheta_1 \sin 2\theta \right) \sin 2 (\vartheta_2 + \theta),
\]

(5.51)

\[
\mathbb{R}_{P-P}(\vartheta_1, \vartheta_2, \theta, \Delta C'^R_{13}) = \left( \sin^2 \vartheta_1 \sin^2 \theta + \cos^2 \vartheta_1 \cos^2 \theta + \frac{1}{2} \sin 2\vartheta_1 \sin 2\theta \right) \sin^2 (\vartheta_2 + \theta) + \left( \sin^2 \vartheta_1 \sin^2 \theta + \cos^2 \vartheta_1 \cos^2 \theta - \frac{1}{2} \sin 2\vartheta_1 \sin 2\theta \right) \cos^2 (\vartheta_2 + \theta),
\]

(5.52)

\[
\mathbb{R}_{P-SV}(\vartheta_1, \vartheta_2, \theta, \Delta C'^R_{13}) = \frac{1}{2} \left[ \left( \sin^2 \vartheta_1 \sin^2 \theta + \cos^2 \vartheta_1 \cos^2 \theta + \frac{1}{2} \sin 2\vartheta_1 \sin 2\theta \right) \sin 2 (\vartheta_2 + \theta) \right. \\
- \left. \left( \sin^2 \vartheta_1 \sin^2 \theta + \cos^2 \vartheta_1 \cos^2 \theta - \frac{1}{2} \sin 2\vartheta_1 \sin 2\theta \right) \right] \sin 2 (\vartheta_2 + \theta),
\]

(5.53)

\[
\mathbb{R}_{P-P}(\vartheta_1, \vartheta_2, \theta, \Delta C'^R_{15}) = \left( \sin 2\theta + \sin 2\vartheta_1 \cos 2\theta \right) \sin^2 (\vartheta_2 + \theta) + \left( \sin^2 \vartheta_1 \cos^2 \theta + \cos^2 \vartheta_1 \sin^2 \theta - \frac{1}{2} \sin 2\vartheta_1 \sin 2\theta \right) \sin 2 (\vartheta_2 + \theta),
\]

(5.54)

\[
\mathbb{R}_{P-SV}(\vartheta_1, \vartheta_2, \theta, \Delta C'^R_{15}) = \frac{1}{2} \left( \sin 2\theta + \sin 2\vartheta_1 \cos 2\theta \right) \sin 2 (\vartheta_2 + \theta) + \left( \sin^2 \vartheta_1 \cos^2 \theta + \cos^2 \vartheta_1 \sin^2 \theta - \frac{1}{2} \sin 2\vartheta_1 \sin 2\theta \right) \cos 2 (\vartheta_2 + \theta),
\]

(5.55)
\[ R_{P-P} \left( \vartheta_1, \vartheta_2, \theta, \Delta C'_{33} \right) = \left( \sin^2 \vartheta_1 \sin^2 \theta + \cos^2 \vartheta_1 \cos^2 \theta + \frac{1}{2} \sin 2 \vartheta_1 \sin 2 \theta \right) \cos^2 (\vartheta_2 + \theta), \]

\[ R_{P-SV} \left( \vartheta_1, \vartheta_2, \theta, \Delta C'_{33} \right) = -\frac{1}{2} \left( \sin^2 \vartheta_1 \sin^2 \theta + \cos^2 \vartheta_1 \cos^2 \theta + \frac{1}{2} \sin 2 \vartheta_1 \sin 2 \theta \right) \sin 2 (\vartheta_2 + \theta), \]

\[ R_{P-P} \left( \vartheta_1, \vartheta_2, \theta, \Delta C'_{35} \right) = \left( \sin 2 \theta + \sin 2 \vartheta_1 \cos 2 \theta \right) \cos^2 (\vartheta_2 + \theta) \]

\[ + \left( \sin^2 \vartheta_1 \sin^2 \theta + \cos^2 \vartheta_1 \cos^2 \theta + \frac{1}{2} \sin 2 \vartheta_1 \sin 2 \theta \right) \sin 2 (\vartheta_2 + \theta), \]

\[ R_{P-SV} \left( \vartheta_1, \vartheta_2, \theta, \Delta C'_{35} \right) = -\frac{1}{2} \left( \sin 2 \theta + \sin 2 \vartheta_1 \cos 2 \theta \right) \sin 2 (\vartheta_2 + \theta) \]

\[ + \left( \sin^2 \vartheta_1 \sin^2 \theta + \cos^2 \vartheta_1 \cos^2 \theta + \frac{1}{2} \sin 2 \vartheta_1 \sin 2 \theta \right) \cos 2 (\vartheta_2 + \theta), \]

\[ R_{P-P} \left( \vartheta_1, \vartheta_2, \theta, \Delta C'_{55} \right) = \frac{1}{2} \left( \sin 2 \theta + \sin 2 \vartheta_1 \cos 2 \theta \right) \sin 2 (\vartheta_2 + \theta), \]

\[ R_{P-SV} \left( \vartheta_1, \vartheta_2, \theta, \Delta C'_{55} \right) = \left( \sin 2 \theta + \sin 2 \vartheta_1 \cos 2 \theta \right) \cos 2 (\vartheta_2 + \theta), \]

where \( \vartheta_1 \) and \( \vartheta_2 \) are the inclination angles of incident and scattered waves, respectively.

Figure 5.1 shows analytical radiation patterns of a virtual source parameterized in terms of density-normalized Voigt parameters with an incidence angle \( \vartheta_1 \) of 350° and a tilt angle \( \theta \) of 20°. Small values of \( \vartheta_2 \) (< 30°) are related to reflected waves recorded at near offset, where their short wavelengths dominate the inversion. Wide values of \( \vartheta_2 \) (> 330°) are related to reflected and transmitted (diving) waves recorded at far offsets, where their long wavelengths mainly contribute to the inversion. Intermediate \( \vartheta_2 \) values (30°–60° or 300°–330°) are associated with moderate-offset reflected waves. Overall, the radiation patterns of \( C'_{13}, C'_{33}, \) and \( C'_{35} \) are larger than the others. \( C'_{13} \) is well recovered from intermediate-to-wide-angle reflected \( P-P \) and small-to-intermediate-angle reflected \( P-SV \). \( C'_{33} \) is well recovered from small-to-intermediate-
angle reflected $P$-$P$ and $P$-$SV$. $C_{35}^R$ is well recovered from small-to-intermediate-angle reflected $P$-$P$ and small-to-wide angle $P$-$SV$. Trade-offs between $C_{11}^R$ and $C_{13}^R$ and between $C_{35}^R$ and $C_{55}^R$ are expected at intermediate-to-wide angle. Note that different incident and tilt angles show different radiation patterns.

Partial derivatives of the synthesized wavefield with respect to a particular model parameter show the wavefield scattered by a perturbation of that parameter, while the other parameters remain fixed at their background values. Figure 5.2 shows the norm of two-component scattered-wavefields, $\| \frac{\partial u}{\partial m} \|$, for the Voigt parameterization; results for the Chen & Tromp parameterization are very similar and are omitted for brevity. A force source and point diffractor are located at $(x, z) = (1.74$ km, 0.5 km) and (2 km, 2 km), respectively. A homogeneous background model is defined by $C_{11}^R = 13.3427$ km$^2$/s$^2$, $C_{13}^R = 6.5154$ km$^2$/s$^2$, $C_{15}^R = 0.9920$ km$^2$/s$^2$, $C_{33}^R = 9.7340$ km$^2$/s$^2$, $C_{35}^R = 0.5220$ km$^2$/s$^2$, and $C_{55}^R = 2.4286$ km$^2$/s$^2$, and model perturbations, defined by taking 5% of each background value, are $\Delta C_{11}^R = 0.6671$ km$^2$/s$^2$, $\Delta C_{13}^R = 0.3258$ km$^2$/s$^2$, $\Delta C_{15}^R = 0.0496$ km$^2$/s$^2$, $\Delta C_{33}^R = 0.4867$ km$^2$/s$^2$, $\Delta C_{35}^R = 0.0261$ km$^2$/s$^2$, and $\Delta C_{55}^R = 0.1214$ km$^2$/s$^2$. The scattered wavefields $\frac{\partial u}{\partial m}$, representing the interaction of temporal frequency, displacement wavefields, wavespeeds, and radiation patterns (Pan et al., 2016), are calculated in the finite difference sense (Operto et al., 2013).

In Figure 5.2 each parameter perturbation has an influence on $P$-$P$ and $P$-$SV$ diffractions. The amplitude variations of $P$-$SV$ are larger than those of $P$-$P$ depending on the input Voigt parameters. Perturbations in $C_{33}^R$, $C_{35}^R$, and $C_{15}^R$ produce comparable $P$-$P$ diffractions at small, intermediate, and wide scattering angles, respectively, which are larger than in the other parameters, as indicated by the analytical radiation patterns. $P$-$P$ diffraction of $C_{55}^R$ is large at intermediate angles, and larger than $C_{15}^R$, which is large at wide angle. For the case where $\vartheta_1 = 350^\circ$ and
\( \theta = 20^\circ \), a perturbation in \( C^{R}_{11} \) produces large \( P-P \) diffraction at wide angles and the weakest of the other parameters, making its model update small.

### 5.3 Methodology

To carry out forward and adjoint simulations with isotropic acoustic wave propagation in the water layer coupled with anisotropic elastic wave propagation below, we used the open source SPECFEM2D package (Komatitsch and Vilotte, 1998a; Komatitsch and Tromp, 1999; Komatitsch et al., 2000b,a). Because we are unaware of any acceptable open source finite difference implementations for TTI simulations, the observed and synthetic seismograms are both generated using a spectral-element method. Our spectral-element mesh consisted of 3,600 elements and 90,000 global grid points and fully honored the bathymetry. The unstructured quadrilateral spectral-element mesh with an average size of 100 m is defined using \( 5 \times 5 \) Gauss-Legendre-Lobatto (GLL) points. The size of the model was 10.68 km across and 3.71 km in depth. A free surface boundary condition was used at the top and absorbing boundary conditions at the remaining sides.

For the inversion experiments, data are simulated from 64 shots 5 m below the water surface, with a 5 Hz dominant frequency Ricker source-time function and a 155 m lateral spacing between shots. An ocean-bottom geometry is used for data acquisition, with 502 fixed two-component geophones and a 20 m lateral spacing along the sea floor.

During the inversions, properties of the water column are presumed known and kept fixed. Density updates are computed via Gardner’s equation from the Fréchet derivatives with respect to vertical \( P \)-wavespeed. A standard FWI technique, without any type of multiscale continuation, is performed. Model parameters are updated via

\[
m_{n+1} = m_n - \alpha_n H_n^{-1} g_n, \tag{5.62}
\]
where $\alpha$ denotes the step length, $H^{-1}$ the L-BFGS approximation to the inverse Hessian (Nocedal and Wright, 2006b), and $g$ the gradient. The step length $\alpha_n$ is calculated by a line search, using Algorithm 3.5 in Nocedal and Wright (2006b). Before applying the L-BFGS inverse Hessian, the gradient vector is smoothed by a 2D Gaussian function with a 50 m standard deviation.

5.4 Numerical experiments

5.4.1 Example 1: Inverting for vertical wavespeeds and Thomsen parameters

We start the numerical experiments by recovering vertical wavespeeds and Thomsen parameters for VTI, which is a specific case of TTI where $\theta = 0^\circ$. The variation in the misfit function is expressed as

$$\delta \chi = \int (\delta \ln \rho K'_\rho + \delta \ln \alpha_0 K_{\alpha_0} + \delta \ln \beta_0 K_{\beta_0} + \delta \ln \varepsilon K_\varepsilon + \delta \ln \delta K_\delta) \, d^3x, \quad (5.63)$$

where

$$K'_\rho = \rho K_\rho + \rho \alpha_0^2 [K_{c11} (1 + 2\varepsilon) + K_{c13} (1 + 2\delta) + K_{c33}] + \rho \beta_0^2 [K_{c55} - 2K_{c13}], \quad (5.64)$$

$$K_{\alpha_0} = 2\rho \alpha_0^2 [K_{c11} (1 + 2\varepsilon) + K_{c13} (1 + 2\delta) + K_{c33}], \quad (5.65)$$

$$K_{\beta_0} = 2\rho \beta_0^2 [K_{c55} - 2K_{c13}], \quad (5.66)$$

$$K_\varepsilon = 2\rho \alpha_0^2 \varepsilon K_{c11}, \quad (5.67)$$

$$K_\delta = 2\rho \alpha_0^2 \delta K_{c13}. \quad (5.68)$$
Equation (5.63) is written in terms of relative perturbations, for example, $\delta \ln \rho = \delta \rho / \rho$ is the relative perturbation in mass density. Equations (5.64)–(5.68) are sensitivity kernels of impedance, vertical $P$-, and $S$-wavespeeds, $\varepsilon$, and $\delta$, respectively. For the general TTI case, an inversion involving the Thomsen parameters and a tilt angle is difficult, because the angle can vary widely; this is what motivates our inversions in terms of the Voigt and Chen & Tromp parameters.

Figure 5.4 shows a 2D target model which consists of density $\rho$, vertical $P$- and $S$-wavespeeds, $\alpha_0$ and $\beta_0$, Thomsen parameters, $\varepsilon$ and $\delta$, and tilt angle cross sections. From the BP TTI model, density is obtained using Gardner’s relationship (Gardner et al., 1974), $\rho = 0.31 \alpha_0^{0.25}$, and vertical $S$-wavespeed is obtained by scaling vertical $P$-wavespeed by a factor of 0.55. The tilt-angle parameter represents the orientation, measured clockwise from vertical, of the anisotropic symmetry axis perpendicular to bedding plane of the geological structure.

Figure 5.3 shows the initial model in terms of vertical wavespeeds, $\alpha_0$ and $\beta_0$, and Thomsen parameters, $\varepsilon$ and $\delta$. The initial model parameters are obtained by convolving the corresponding true parameters in Figure 5.4 with a 2D Gaussian function with a 700 m standard deviation. The tilt angle is not used in this example.

Figure 5.16 shows the corresponding inverted parameters after 81 iterations. The wavespeeds $\alpha_0$ and $\beta_0$ are nicely recovered. The inverted $\varepsilon$ and $\delta$ are not as well constrained as the other parameters, which are due to trade-offs with $\alpha_0$. Our analysis of the inversion result is consistent with several other studies (e.g., Operto et al., 2013; Gholami et al., 2013b).
5.4.2 Example 2: Inverting for Voigt and Chen & Tromp parameters

In this example, the Voigt or Chen & Tromp parameters are used to describe 2D TTI. The wavespeeds, Thomsen parameters, and tilt angle are well recovered from the corresponding inverted models.

Figures 5.5 and 5.6 show the target model in terms of density-normalized Voigt parameters obtained via equations (5.21)–(5.26) and Chen & Tromp parameters obtained via equations (5.29)–(5.34), respectively. Besides wavespeed information, the off-diagonal Voigt parameters $C_{15}^{\prime R}$ and $C_{35}^{\prime R}$ and the associated Chen & Tromp parameters $J^\prime$ and $K^\prime$ provide information about the orientation of the anisotropic symmetry axis.

Figure 5.7 shows representative data generated by a force source located at a position of 5.27 km recorded by ocean-bottom geophones. Seismograms are plotted with the same amplitude scale and contain first-arrival, multiple, and reflected waves. The reflections do not appear very strong because they are overshadowed by the first arrivals. Overall, the reflection data used in the inversion vary in offsets from $0^\circ$ to $\sim 68^\circ$; the model updates are dominated by pre-critical reflections. Time-reversed waveform differences between synthetics and data, which are normalized by equation (5.6), are used as adjoint sources in the gradient computation (Zhu et al., 2009). The normalization makes the Fréchet derivatives independent of source magnitude.

Finally, Figures 5.8 and 5.9 show the initial model in terms of density-normalized Voigt and Chen & Tromp parameters. The initial Voigt parameters $C_{11}^{\prime R}$, $C_{13}^{\prime R}$, $C_{33}^{\prime R}$, and $C_{55}^{\prime R}$ are obtained by convolving the corresponding true parameters in Figure 5.5 with a 2D Gaussian function with a 650 m standard deviation. The anticlinal structure present in the true model has been smoothed out in the initial model. When Voigt parameter are used, a reasonable non-zero starting estimate for $C_{15}^{\prime R}$ and $C_{35}^{\prime R}$ is required to ensure convergence. In our case, the initial $C_{15}^{\prime R}$ and $C_{35}^{\prime R}$ are obtained by
scaling down the corresponding true parameters by a factor of 0.025. The minimum and maximum values of $C_{15}^{tR}$ are $-0.0344 \text{ km}^2/\text{s}^2$ and 0.0386 $\text{km}^2/\text{s}^2$, respectively. The minimum and maximum values of $C_{35}^{tR}$ are $-0.0383 \text{ km}^2/\text{s}^2$ and 0.0372 $\text{km}^2/\text{s}^2$, respectively. The near-zero values of $C_{15}^{tR}$ and $C_{35}^{tR}$ indicate that the initial model is essentially VTI. The initial Chen & Tromp parameters are obtained from the initial Voigt parameters via equations (5.29)–(5.34).

**Inversion results**

Figure 5.10 shows representative horizontal and vertical displacement traces corresponding to a shot located at $x = 5.27 \text{ km}$ in the initial Voigt model, and corresponding vertical and horizontal data residual traces. The RMS residual between synthetic displacements calculated in the initial model and observed displacements is $\sim 98.1\%$. This indicates that the initial model is sufficiently far from the true model. The large contributions come mainly from middle-to-far offset data residuals. Synthetic displacements produced by the initial Chen & Tromp model and its data residual traces are the same as those for the initial Voigt model.

Corresponding to the Voigt and Chen & Tromp parameterizations, Figures 5.11 and 5.12 show the inverted models after 55 and 60 iterations, respectively. The inverted Voigt and Chen & Tromp parameters after 55 and 60 iterations, respectively, do not change much. With both the Voigt and Chen & Tromp parameterizations, recovery of the geological structure and the orientation of the anisotropic symmetry axis is good. The structural dips, which rotate the radiation patterns toward small or intermediate scattering angles, respectively, provide additional constraints on $C_{13}^{tR}$.

Depth profiles of the initial (blue line), inverted (red line), and true (black line) models extracted at $x = 6.26 \text{ km}$ are shown in Figures 5.13 and 5.14. The initial values of $C_{11}^{tR}$, $C_{13}^{tR}$, $C_{33}^{tR}$, and $C_{55}^{tR}$, and $A'$, $C'$, $L'$, and $F'$ increase smoothly with depth.
The initial values of $C'_{15}$ and $C'_{35}$, and $J'$ and $K'$ are close to zero. Correspondence between the true and inverted depth profiles is good.

Figure 5.15 shows a model consisting of density $\rho$, vertical $P$- and $S$-wavespeeds, $\alpha_0$ and $\beta_0$, and the Thomsen parameters, $\varepsilon$ and $\delta$, and tilt angle $\theta$, which are obtained from the inverted Voigt parameters. The vertical wavespeeds and Thomsen parameters are estimated by using the inverse relationships (5.10)–(5.13). Density is estimated using Gadner’s relationship. The VTI parameters, $C'_{11}$, $C'_{13}$, $C'_{33}$, and $C'_{55}$ in equations (5.10)–(5.13), are recovered by solving equation (5.49) using Weighted Least Squares (Menke, 2012) with the weighting matrix

$$W_e = \begin{pmatrix} 0.01 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.01 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0.01 & 0 & 0 & 1 & 0 & 0 & 0.01 \\ 0.01 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0.01 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}. \tag{5.69}$$

The tilt angle is estimated using equation (5.45). The corresponding model estimated from the inverted Chen & Tromp parameters is shown in Figure 5.16. The vertical wavespeeds and tilt angle are well estimated. The Thomsen parameters $\varepsilon$ and $\delta$ show trade-offs with $\alpha_0$, and are not as well reconstructed as the other parameters. The tilt angle, $\varepsilon$, and $\delta$ obtained from the Voigt parameters are more accurate than those obtained from the Chen & Tromp parameters. The estimated vertical wavespeeds and Thomsen parameters may be improved by an iterative inversion procedure. Using a parameterization of 2D seismic anisotropy in terms of the Voigt or Chen & Tromp parameters, the estimated vertical wavespeeds, Thomsen parameters, and tilt angle are more accurately recovered than in previous studies (e.g., Oropeza and McMechan, 2014).
Figures 5.17 shows the anisotropic fast axes estimated from the initial, inverted, and true models. By convention, the orientation of the fast axis is perpendicular to the orientation of the anisotropic symmetry axis calculated using equation (5.45). In fast axis vector plots, the lengths of the dashes are calculated using equation (5.47), which provides a measure of the strength of the anisotropy. The initial fast axis is essentially VTI following the flat layering and getting larger with depth. The inverted fast axis is similar to the true TTI fast axis.

Differences between the fast axes estimated from the initial and true models, and the fast axes estimated from the inverted and true models, using the Voigt parameterization, are shown in Figure 5.18. The RMS residual of between the true and initial fast axes is $\sim 70.2\%$, and that between the true and inverted fast axes is $\sim 8.3\%$.

Figure 5.19 shows the fast axis estimated from the inverted Chen & Tromp parameters after 60 iterations, and corresponding difference between fast axes estimated from the inverted and true models. The RMS residual of the fast axis is $\sim 9.3\%$, which is $\sim 8\times$ smaller than the fast axes estimated from the initial and true models. The estimated fast axes based on the Voigt and Chen & Tromp parameters, respectively, in Figures 5.17b and 5.19a are similar to the true fast axes in Figure 5.17c.

Figures 5.20 shows representative synthetic displacement traces computed from the inverted Voigt parameters and corresponding data residuals, and the data residuals in the Chen & Tromp inverted model. The RMS residual of the synthetic displacements which are produced by the inverted Voigt parameters, and data displacements is $\sim 5.7\%$, which is relatively small compared to the observed data. The residual of the horizontal displacement is larger than that of the vertical displacement. The RMS residual of the synthetic which are displacements calculated by the inverted Chen & Tromp parameters, and data displacements is $\sim 8.6\%$. The RMS residual of the Chen & Tromp parameters is slightly higher than that of the Voigt parameters.
Figure 5.21 compares the convergence rates for the Voigt (blue line) and Chen & Tromp (red line) parameterizations. The rate of convergence for the Chen & Tromp parameters is slightly slower than for the Voigt parameters, but the end results, represented by the fast axes in Figures 5.17b and 5.19a, are comparable. The fast axis vector, capturing the anisotropic strength and tilt angle, shows the anisotropy distribution more informatively (e.g., Bachrach, 2015).

Imaging results

Figure 5.22 shows impedance kernels, $K'_\rho$, for the Voigt parameterization at various iterations. Low-frequency artifacts which commonly exist in migration with the two-way wave-equation are significantly reduced and the resolution of the seismic images increases when the inverted models get closer to the solution. Water multiples are eliminated by subtracting synthetics which are calculated from a model consisting of acoustic (water) and elastic (solid) domains with the right bathymetry from the corresponding data. As the inversion converges, the model parameters become more precise and the impedance kernel becomes smaller. The impedance kernel is crisp and very promising for anisotropic seismic imaging.

5.4.3 Example 3: Initial model with a small constant $\delta$

In surface seismic acquisition, the Thomsen parameter $\delta$ is inaccurately known. In this experiment, the initial Voigt parameters are calculated using a constant $\delta$ of 0.01. This $\delta$ is small and relatively far from the average true $\delta$ shown in Figure 5.4e. Using this $\delta$ in equation (5.11) and substituting equations (5.10)–(5.13) into equations (5.21)–(5.26), the Voigt parameters are obtained. The initial Voigt parameters used for this experiment (Figure 5.23) are a smooth version of the true Voigt parameters. Note that the initial $C'_{13}$ in Figure 5.23b is smaller than the value used in the previous
experiment (Figure 5.8b). The same target model used in the previous experiment (Figure 5.5) is used for calculating the observed data.

Figure 5.24 shows the inverted model of density-normalized Voigt parameters after 63 iterations using the initial model with a constant \( \delta \) of 0.01. \( C'_{11}, C'_{13}, C'_{33}, \) and \( C'_{55} \) are well recovered and comparable to the values in Figure 5.11. \( C'_{15} \) and \( C'_{35} \) are slightly smaller than those in Figure 5.11 but still reasonably well recovered.

### 5.4.4 Example 4: True model calculated with \( \beta_0 \) independent of \( \alpha_0 \)

In complex geological structures, the model parameters do not always have the same tendency of increasing with increasing depth. To imitate this behavior, the true S-wavespeed \( \beta_0 \) in Figure 5.4b will be replaced by the substitution

\[
\beta_0 \rightarrow - \beta_0 + 2\beta_{0sm} + 150, \tag{5.70}
\]

where \( \beta_{0sm} \) is a smooth version of \( \beta_0 \) in Figure 5.4b. Here, \( \alpha_0 \) and the new \( \beta_0 \) no longer look the same because the new \( \beta_0 \) is not directly scaled from \( \alpha_0 \). Using this \( \beta_0 \) in equation (5.11) and substituting equations (5.10)–(5.13) into equations (5.21)–(5.26) the Voigt parameters are obtained, which are shown in Figure 5.27.

\( C'_{11} \) and \( C'_{33} \) have the same tendency of increasing with increasing depth. In contrast, \( C'_{13} \) and \( C'_{55} \) do not always increase with increasing depth. \( C'_{13} \) and \( C'_{55} \) are different from those in Figures 5.5b and f, and \( C'_{15} \) and \( C'_{35} \) are similar to those in Figures 5.5c and e. \( C'_{33} \) and \( C'_{55} \) no longer look the same.

Figure 5.28 shows the initial model of density-normalized Voigt parameters obtained by convolving the true model with a 2D Gaussian function with a 650 m standard deviation. The anticlinal structure present in the true model has been smoothed out in the initial model.
Figure 5.29 shows the inverted model of density-normalized Voigt parameters after 77 iterations. $C_{11}^{\prime R}$, $C_{13}^{\prime R}$, $C_{33}^{\prime R}$, and $C_{55}^{\prime R}$ are well retrieved. There is no significant cross-talk between $C_{33}^{\prime R}$ and $C_{55}^{\prime R}$. Although $C_{13}^{\prime R}$ and $C_{55}^{\prime R}$ have a reverse tendency to $C_{11}^{\prime R}$ and $C_{33}^{\prime R}$, the recovery of $C_{15}^{\prime R}$ and $C_{35}^{\prime R}$ is fairly accurate.

### 5.5 Conclusions

Starting from a VTI model with little knowledge of the geologic structure, we have shown that a full TTI target model, including anisotropic direction and strength, can be recovered. By avoiding explicit use of a rotation angle parameter, the Voigt or Chen & Tromp parameters allow accurate and efficient recovery.
Figure 5.1: Analytical radiation patterns, $R_{P-P}$ and $R_{P-SV}$, for a virtual source parameterized in terms of density-normalized Voigt parameters with incidence angle $\theta_1 = 350^\circ$ and tilt angle $\theta = 20^\circ$. Scattering modes, $P-P$ and $P-SV$, are indicated by the blue and red lines, respectively. Model parameters and scattering angle $\vartheta_2$ are labeled.
Figure 5.2: Norm of two-component scattered-wavefields, $\| \frac{\partial u}{\partial m} \|$, parameterized in terms of density-normalized Voigt parameters. The source and diffractor are located at $(x, z) = (1.74 \text{ km}, 0.5 \text{ km})$ and $(2 \text{ km}, 2 \text{ km})$ shown by the red star and blue dot, respectively. Model parameters, scattering modes, and scales of amplitude variation are labeled.
Figure 5.3: Initial model in terms of vertical $P$-wavespeed $\alpha_0$, vertical $S$-wavespeed $\beta_0$, and Thomsen parameters, $\varepsilon$ and $\delta$.

Figure 5.4: Target model in terms of mass density $\rho$, vertical $P$-wavespeed $\alpha_0$, vertical $S$-wavespeed $\beta_0$, Thomsen parameters, $\varepsilon$ and $\delta$, and tilt angle $\theta$. 
Figure 5.5: Target model in terms of density-normalized Voigt parameters.

Figure 5.6: Target model in terms of density-normalized Chen & Tromp parameters.
Figure 5.7: Representative (a) horizontal and (b) vertical displacement traces corresponding to a shot located at $x = 5.27$ km in the target model.
Figure 5.8: Initial model in terms of density-normalized Voigt parameters.

Figure 5.9: Initial model in terms of density-normalized Chen & Tromp parameters.
Figure 5.10: Representative (a) horizontal and (b) vertical displacement traces corresponding to a shot located at $x = 5.27$ km in the initial Voigt model. Corresponding vertical and horizontal data residual traces are shown in panels (c) and (d), respectively.
Figure 5.11: Inversion result after 55 iterations obtained using density-normalized Voigt parameters.

Figure 5.12: Inversion result after 60 iterations obtained using density-normalized Chen & Tromp parameters.
Figure 5.13: Voigt parameter depth profiles from the initial (blue line), inverted (red line), and true (black line) models extracted at $x = 6.26$ km.
Figure 5.14: Chen & Tromp parameter depth profiles from the initial (blue line), inverted (red line), and true (black line) models extracted at $x = 6.26$ km.
Figure 5.15: Inverted model in terms of mass density $\rho$, vertical $P$-wavespeed $\alpha_0$, vertical $S$-wavespeed $\beta_0$, Thomsen parameters, $\varepsilon$ and $\delta$, and tilt angle $\theta$, estimated from the inverted Voigt parameters.

Figure 5.16: Inverted model in terms of mass density $\rho$, vertical $P$-wavespeed $\alpha_0$, vertical $S$-wavespeed $\beta_0$, Thomsen parameters, $\varepsilon$ and $\delta$, and tilt angle $\theta$, estimated from the inverted Chen & Tromp parameters.
Figure 5.17: Fast axes estimated from the (a) initial, (b) inverted, and (c) true models obtained using density-normalized Voigt parameters. The fast axis of the initial model is VTI and gets larger with depth. The fast axis of the true model follows the geological structure and overall gets larger with depth.
Figure 5.18: Error in estimated fast axes. Panel (a) shows the difference between fast axes from the initial and true models. Panel (b) shows the difference between fast axes from the Voigt inverted model and true model.

Figure 5.19: Fast axis estimated from the (a) inverted model obtained using density-normalized Chen & Tromp parameters and (b) corresponding estimation error.
Figure 5.20: Vertical and horizontal displacement traces in panels (a) and (b), respectively, based on a shot located at \( x = 5.27 \) km in the Voigt inverted model after 55 iterations. Corresponding vertical and horizontal data residual traces in panels (c) and (d), respectively. Vertical and horizontal data residual traces in the Chen & Tromp inverted model after 60 iterations in panels (e) and (f), respectively.
Figure 5.21: Convergence rate for Voigt (blue line) and Chen & Tromp (red line) inversions. Data misfit is plotted against the number of iterations.
Figure 5.22: Imaging results obtained using the Voigt model impedance kernel after (a) 10, (b) 25, and (c) 40 iterations. The seismic images gradually improve and sharpen as the inverted models approach the true solution.
Figure 5.23: Initial model in terms of density-normalized Voigt parameters with a constant $\delta$ of 0.01.

Figure 5.24: Inversion result after 63 iterations obtained using the initial model shown in Figure 5.23.
Figure 5.25: Target model in terms of density-normalized Voigt parameters calculated with a $\beta_0$ that is independent of $\alpha_0$.

Figure 5.26: Initial model in terms of density-normalized Voigt parameters calculated with true $\beta_0$ independent of $\alpha_0$. 
Figure 5.27: Target model in terms of density-normalized Voigt parameters calculated with a $\beta_0$ that is independent of $\alpha_0$.

Figure 5.28: Initial model in terms of density-normalized Voigt parameters calculated with true $\beta_0$ independent of $\alpha_0$. 
Figure 5.29: Inversion result after 77 iterations obtained using the initial model shown in Figure 5.28.
Chapter 6

SeisFlows: a flexible and portable waveform inversion package

SeisFlows is an open source Python package that delivers a complete, customizable waveform inversion workflow and framework for research in earthquake tomography, oil and gas exploration, and medical imaging. New methods can be rapidly prototyped in SeisFlows by modifying default inversion or migration classes, and through support for high-performance systems and massively parallel solvers, strategies tested on inexpensive 2D problems can be subsequently applied to 3D inversion with thousands of seismic traces and billions of model parameters. Besides running very small inversions on individual laptops and desktops, SeisFlows has, to date, run on supercomputers managed by the Department of Defense, Chevron Corp., Total S.A. and Princeton University.

6.1 Introduction

Whether applied to a human body, a hydrocarbon reservoir, or an entire continent, waveform inversion is a complicated and computationally expensive procedure. If
inversion software is to be of use to a wide community and remain relevant in an evolving field it must satisfy a number of requirements.

First, it must be flexible. Each of the component parts must be designed in a way that allows for modification or extension. Wave-equation solvers, nonlinear optimization algorithms, and signal processing toolkits must be made to work together through carefully thought-out interfaces.

Second, a package must be capable of running in a variety of hardware and software environments. For research, being able to prototype on a laptop, desktop or small cluster is desirable. For large-scale, high-resolution 3D inversions, being able to run massively parallel simulations on a peta-scale cluster is a prerequisite. Portability can be especially difficult at the high-performance end because many different cluster configurations exist with little standardization between them.

To manage this complexity and meet these challenges, the waveform inversion problem is broken down and abstracted into six component parts: (1) solver, (2) system, (3) preprocessing, (4) postprocessing, (5) nonlinear optimization and (6) workflow. The source code underlying SeisFlows is structured in a modular way based on these categories. Users are offered various choices for each category. For example, if the study area in an earthquake tomography project expands, users can trade a Cartesian near-surface solver for a spherical whole-earth solver. If a PBS cluster goes offline and a SLURM cluster comes online to replace it, users can trade the PBS system interface for a SLURM system interface. If desired functionality is missing from the main package, users can contribute their own classes or overload default ones.

This chapter is divided into four sections. First, we elaborate on the flexibility and portability goals motivating SeisFlows. Second, we describe each of the six component parts and how they work together to meet the design goals. Third, we illustrate the use of the package in a high-performance computing environment through a challenging and computationally expensive near-surface inversion example. In the final
section, we provide an introduction to practical use of the package to supplement the tutorial in Appendix B.

6.2 Design goals

Before being released to the community, SeisFlows began as a response to difficulties faced by a single research group. About five years ago everyone in the Theoretical and Computational Seismology group at Princeton University was interested in waveform inversion, but the wide variety of applications we were working on and the need for rapidly prototyping new methods in 2D and later applying them in 3D led our codes to become forked and fragmented. This in turn led to duplication of effort between developers, increased susceptibility to bugs, and difficulty bringing new members up to speed with the code base. A framework for streamlining our software development efforts was badly needed.

Besides the need for a flexible research framework, we also faced challenges involving portability. Our local cluster was too small to perform all the expensive 3D inversions we were on working at the time, and other supercomputers were configured differently than our local cluster. When starting out on a new machine, sometimes it took weeks of set up work to get to the point where we could simply run our codes. Although SeisFlows was envisioned originally as a research framework, it became clear that many of the same principles used to provide a flexible research framework could also be used to provide portable production tool across many high-performance computing environments.

Of course, flexibility and portability are important goals not just in scientific projects but in almost all software development. In this section we discuss our design choices concerning these goals before moving on in the next section to a more detailed package overview.
6.2.1 Flexibility

Because codes inevitably grow and evolve, flexibility is an essential requirement in software engineering. A framework in which new functionality can be added without compromising usability or maintainability helps a package remain useful even despite rapidly evolving research ideas or a high-turnover development team.

Software engineering best practices involving modular design and object oriented programming are the key to developing flexible software (e.g., Gamma et al. 1995). By breaking down complex structures into fundamental building blocks, modularity allows one component to be exchanged or modified without disruptive package-wide changes. Similarly, by allowing code reuse via inheritance, object oriented programming provides a minimally disruptive way to customize default classes.

Such principles can be difficult to apply to scientific projects, however, because of lack of programming expertise among domain scientists or difficulties posed by legacy codes. An effective response to these challenges, we found, was to develop a Python workflow framework. Besides providing ease of use for domain practitioners and various options for integrating legacy solvers, Python also provides powerful object oriented capabilities and a growing collection of scientific tools, including the numpy package for numerical work (Walt et al., 2011) and the outstanding obspy package for data processing (Krischer et al., 2015).

6.2.2 Portability

Whether because of the needs of a developer to reach a wide audience or the needs of a researcher to run on different clusters, portability is a common software engineering requirement. For high-performance computing, the problem can be especially daunting because at the high end computational environments are extremely diverse. Compatibility with different processor architectures, filesystems, memory configurations and job schedulers are required for an HPC code to be considered portable.
The approach in SeisFlows to portability is to provide an interface layer through which the workflow interacts with the system resources. To launch a set of forward simulations, for example, the user invokes the forward method of the solver via the run method of the system interface. By isolating environment-dependent attributes, the system interface provides a consistent command set across different computing environments.

This approach also provides a fair amount of scalability. By trading one system interface for another, a user can switch from running simulations in serial on a laptop or desktop to running simulations in parallel on a cluster. Take the system.run and solver.forward methods mentioned above as an example. When running on a laptop or desktop, system.run can be used to carry out the forward simulations in serial within a loop. Alternatively, when running on a cluster, system.run can be used to allocate resources through the scheduler and then execute the forward simulations in parallel via a job array. In case the cluster has one or two bad nodes, fault tolerance can even be provided by checking the exit status of the solver processes and rerunning any failed simulations.

### 6.3 Package organization

As illustrated in Fig. 1, SeisFlows consists of six component. This structure represents the only hardwired aspect of the package; virtually everything else, including the behavior of the individual components, is flexible.

#### 6.3.1 Solver

As the software component used to simulate wave propagation, the solver is the main computational engine underlying an inversion. In the course of a single model update, the solver is first used to generate synthetic data and then to backproject
data residuals (Tromp et al., 2005). The adjoint operator used for backprojection can be based either on an “optimize-then-discretize” approach or a “discretize-then-optimize” approach, as they are commonly referred to in the literature (Gunzburger, 2000).

Choices for acoustic or elastic wave simulation in heterogeneous media include finite-difference, finite-element, and spectral-element numerical schemes. The type of solver best-suited to a particular inversion depends on whether the data involve body waves, surface waves or both; whether the topographic or bathymetric variations are small or large; and whether the material properties vary continuously or discontinuously within the target structure.

While in principle SeisFlows can interface with all the types of solvers mentioned above, to date, only spectral-element solver interfaces are included in the main repository. Interface classes for the SPECFEM2D, SPECFEM3D and SPECFEM3D_GLOBE packages are provided, supporting applications from 2D medical imaging to 3D near-surface subsalt imaging to 3D whole-earth tomography. All solver interfaces inherit from a common base class to avoid code duplication. Recently, some researchers have begun using SeisFlows in combination with finite-difference solvers (Gian Mattharu, personal communication), but such functionality is not available yet in the main package.

6.3.2 System

As the software layer through which the solver and other workflow components interact with the system resources, the “system” component provides interoperability from one environment to another. A selection of interfaces is provided in the main package, including some for laptops and desktops and others for LSF, SLURM and PBS clusters. In all cases, the idea of the system component is to provide workable default configuration that allows for further customization in case anything about the user’s environment requires it.
6.3.3 Preprocessing

In our terminology, “preprocessing” refers to signal processing operations carried out on seismic traces. The name reflects the fact that such operations are usually performed prior to data backprojection.

The default preprocessing class relies on `obspy` for reading and writing data and signal processing. If a file format is not supported by `obspy`, users can contribute their own reading and writing utilities through a simple plugin system.

A variety of data processing options are included in the default class. Choices are provided for highpass, lowpass and bandpass filtering; trace-by-trace or record section-by-record section normalization; and muting early or late arrivals.

Finally, machinery for generating adjoint traces is provided for use with “optimize-then-discretize” solvers. Adjoint trace generators corresponding to waveform difference, travelt ime, envelope and instantaneous phase objective functions are provided. Support for other objective functions can be added, again through a plugin system.

6.3.4 Postprocessing

In our terminology, “postprocessing” refers to imaging processing operations on models, sensitivity kernels, or migrated images. The name reflects the fact that such operations are usually performed after data backprojection.

A wide variety of operations fall into this category. Techniques including smoothing, basis projection, and Tikhonov and total variation regularization all involve operations on sensitivity kernels and are included in the main package. Treatment of site effects or numerical artifacts around sources or receivers also falls under postprocessing, as does spatial filtering or sharpening performed in oil and gas exploration contexts.

Because postprocessing operations are sometimes performed directly on the model or kernels as expressed in the finite-difference, finite-element or spectral element basis
used by the solver, there may be some overlap between the solver and postprocessing components. Rather than reimplementing this functionality in Python, in SeisFlows we use routines included in the SPECFEM2D, SPECFEM3D and SPECFEM3D_GLOBE packages for some postprocessing functions.

6.3.5 Nonlinear optimization

What drives the progress of an inversion, both in terms of generating model updates and reducing data misfit, is the nonlinear optimization procedure. The rate of convergence in waveform inversion depends on the particular nonlinear optimization algorithm chosen.

In the main package, available nonlinear optimization algorithms include gradient descent, nonlinear conjugate gradient (NLCG) and quasi-Newton (QN) algorithms, with bracketing and backtracking line searches and restart safeguards included in the implementation. Default numerical settings should work well for a wide range inversions, but tuning parameters are provided just in case. To improve performance beyond the supralinear convergence offered by NLCG and QN, preconditioners can be loaded through a plugin system. Finally, a nonlinear optimization “integration test” is provided to allow the model update machinery to run using the Rosenbrock function \cite{Rosenbrock1960} or other inexpensive test problems for easy benchmarking or debugging.

6.3.6 Workflow

Finally, the thing that ties everything together is the “workflow.” Default inversion and migration workflows are provided that can be used directly or as a base class on top of which specialized strategies can be implemented.

In practice, execution of a workflow is equivalent to stepping through the code contained in \texttt{workflow.main}. Users are free to modify the default inversion and
migration workflows, which are designed to be easily customizable. For example, by including `initialize` and `finalize` methods that can be used for any necessary setup or cleanup work before or after a model update.

Visual depictions of the default inversion and migration workflows are shown in Fig. 2. Because each component part is highly customizable, only a very schematic representation is provided. To give a sense for the type of domain-specific applications possible through SeisFlows, examples of more specialized whole-earth and near-surface inversion workflows are depicted in Figs. 3 and 4.

Reflecting our group’s use of SeisFlows as a research framework, a number of custom strategies have been implemented by overloading the inversion and migration classes. Ambient noise inversion, stochastic inversion, and double-difference waveform inversion, for instance, could all be implemented in this manner. Finally, in addition to carrying out inversions and migrations, users can script entirely new workflows by invoking the solver, preprocessing, postprocessing, and nonlinear optimization components in any desired sequence. Velocity analysis or uncertainty quantification tasks could be implemented in this way, to give just a few examples.
Figure 6.1: Structure of the SeisFlows package. A selection of main options or settings are listed for each of the six parts that comprise the package.
Figure 6.2: Default inversion and migration workflows provided by SeisFlows. Because each component part is highly customizable, only a very schematic representation is shown above.
Figure 6.3: One example of a domain-specific application that can be implemented in SeisFlows by overloading default classes. Special gradient smoothing operations can be used to address spatially uneven distribution of seismic stations and earthquakes in global seismology. To this end, users can choose from existing options within the postprocess category or provide their own custom postprocessing class.
Prepare Cartesian solver mesh
Filter data

SOLVER (FORWARD)

PREPROCESS
Filter synthetics
Compare data and synthetics
Generate adjoint sources used in "optimize-then-discretize" approach

SOLVER (ADJOINT)

POSTPROCESS
Remove source/receiver artifacts
Sum source contributions
Apply Tikohonov penalty function contribution

NONLINEAR OPTIMIZATION
Initialize quasi-Newton inverse Hessian
Generate quasi-Newton search direction
Perform backtracking line search

Figure 6.4: Another example of a domain-specific application that can be implemented in SeisFlows by overloading default classes. Many specialized data preprocessing operations have been developed for near-surface data, which lack the well-behaved crust, mantle and core phases of whole-earth data. To this end, users can choose from existing options within the preprocess category or provide their own custom preprocessing class.
6.4 High-performance computing example

To illustrate the use of SeisFlows on a problem of significant practical interest and large computational expense, we present a near-surface example relevant to oil and gas exploration.

In onshore seismic exploration, traces recorded at the surface are dominated by high-amplitude dispersive surface waves. Rather than removing such information from the data, surface waves can be used as an additional constraint on shallow structure. Unfortunately, several major obstacles, including difficulties in the numerical simulation of surface waves and problematic effects of near-surface complexity, have so far precluded such uses.

To overcome these obstacles, flexibility is required in forward modeling. Most inversion packages use finite-difference modeling schemes, but such methods are not well-suited to surface wave modeling because the free-surface boundary is not treated with enough accuracy. With SeisFlows this is not a serious problem, since the package provides the flexibility to switch from one type of solver to another.

To make use of surface waves, flexibility is also required in the inverse procedure. Most inversion packages employ waveform-difference measures of fit, but these are not well-suited to surface wave inversions because near-surface complexity greatly exacerbates cycle skipping. Again, SeisFlows provides flexibility to get around this problem. To reduce cycle skipping, envelope-misfit functions have been proposed by Bozdag et al. (2011) and Yuan et al. (2015) among others. As with any novel technique, new objective functions require extensive testing before being ready for routine use. SeisFlows provides a useful framework for such testing. Having carried out hundreds of inexpensive 2D comparisons using SeisFlows, including tests of an envelope-based misfit function shown in chapter 2, we now illustrate use of the package through a computationally expensive 3D envelope misfit-based inversion.
6.4.1 Problem

We present results of an envelope-based inversion of body and surface waves generated from the SEAM II foothills model, one of the most challenging industry benchmarks (Oristaglio, 2012). To reduce the overall cost of the experiment, we select a portion of the original model. The selected volume, shown in Fig. 5, is $7 \times 3.5 \times 3$ km$^3$ in the $x$-, $y$- and $z$-directions, respectively. Working in the 1–15 Hz range, some $10^8$ numerical grid points are required for accurate wave simulation.

Despite cropping the model, the computational load of the inversion still exceeded the capacity of our local cluster. We instead used a 2000 core LSF machine made available by Total S.A. Because the cluster configuration was fairly standard, only minor modification of the default LSF interface provided with SeisFlows was necessary, with changes involving specification of the resource queue and MPI library paths.

Figure 4 shows the foothills model and numerical mesh. Topographic variations of as much as 0.9 km lead to significant scattering, focusing, and defocusing effects (Shin et al., 2013). The target model contains strong variations in $P$- and $S$-wavespeeds and density, with discontinuities in $S$-wavespeed in some places of more than 1 km/s. Both the topography and wavespeed contrasts make finite-difference modeling infeasible, so instead we used a spectral-element solver.

6.4.2 Procedure

For the inversion, we used 72 sources and 2,502 receivers regularly distributed on the surface. The distance between shots is 600 m in the $x$- and $y$-directions, while the distance between receivers is 50 m and 200 m, respectively. Each source represents a force applied in the vertical direction with a Ricker wavelet as a source time function. We use a relatively high frequency band (with a dominant frequency of 6 Hz) and we do not use a frequency multiscale approach in the manner of Bunks et al. (1995)
or Yuan et. al (2015). Further, only the vertical component is used for all receivers. Finally, for the model updates we used L-BFGS, which in SeisFlows comes with numerical safeguards (e.g., Dennis and Schnabel 1996) that help provide stability.

### 6.4.3 Results

First, we used an envelope misfit function to simultaneously invert body and surface waves. After 30 iterations the $S$-wavespeed model is significantly improved in the shallow part (Figs. 6e and 6f). Then we used this result as input for a waveform-difference inversion, again using both body and surface waves. After a further 30 iterations shallow structure is improved even more (Figs. 6g and 6h). Having at this point recovered the shallow part of the model using body and surface waves, a conventional body wave-only waveform-difference inversion could be used to improve recovery at depth.

Source records for the shot located at $x=1\text{km}$, $y=1.75\text{km}$ are shown in Fig. 7. As the shallow part of the true model contains strong heterogeneities, the observed data are dominated by large-amplitude, dispersive Rayleigh waves. In contrast, the surface waves in the initial synthetics are much less dispersive because the starting model is relatively smooth. It is clear that the synthetic shot gather generated on a model inverted using envelope-based inversion agrees much more closely with the observed one.

Having inverted body and surface waves generated from a complex near-surface model, this section illustrates the successful use of SeisFlows on a problem that poses significant challenges both in terms of computational cost and environment and in terms of research and methodology.
Figure 6.5: Near-surface seismic inversion example used to illustrate SeisFlows. (a) Target $P$-wavespeed model. (b) Numerical mesh. Adding to the challenge and computational expense of the inversion, the target model contains large topographic variations and complex folded and faulted structures. The unstructured spectral-element numerical mesh contains on the order of $10^8$ integration points.
Figure 6.6: Near-surface seismic inversion results. Vertical slices (left) and horizontal slices (right). (a,b) true model; (c,d) initial model; (e,f) result of envelope-difference inversion; (g,h) result of envelope-difference inversion followed by waveform-difference inversion.
Figure 6.7: A shot record of the SEAM Phase II foothills model, vertical component of velocity. *left:* observed traces, *center:* initial synthetics, *right:* synthetics after envelope FWI.
6.5 Usage overview

6.5.1 Prerequisites

SeisFlows requires Python 2.7, numpy, scipy and obspy. Users will need to install these packages before being able to use SeisFlows.

Access to a computer cluster is required for large-scale inversions. Base classes are provided for several common cluster configurations, including PBS and SLURM. Nonstandard configurations can often be accommodated through modifications to one of the base classes.

6.5.2 Job Submission

Each job must have its own ‘working directory’ within which users must supply two input files, paths.py and parameters.py.

To begin executing a workflow, simply type sfrun within a working directory. If an inversion workflow and serial system configuration, for example, are specified in the parameters file, the inversion will begin executing immediately in serial. If a PBS, SLURM, or LSF system configuration is specified instead, execution may wait until required resources become available.

Once the workflow starts running, status information is displayed to the terminal or to the file output.log. By default, updated models and other inversion results are output to the working directory.

To get a sense for how it all works, try following the step by step “reproducibility” instructions included in the appendix.
6.5.3 Solver Configuration

SeisFlows includes Python interfaces for SPECFEM2D, SPECFEM3D, and SPECFEM3D_GLOBE. While the Python interfaces are part of the SeisFlows package, the solver source code must be downloaded separately through GitHub.

After downloading the solver source code, users must configure and compile it, following the instructions in the solver user manual. Summarized briefly, the configuration and compilation procedure is:

Prior to compilation, users need to run the configure script and prepare input files described in the solver user manual.

To successfully run the configure, you may need to install compilers, libraries, and other software in your environment.

The result of compilation is a set of binary executables including a mesher, solver and various utilities. Note that if solver input files change, solver executables may need to be recompiled.

After compilation, solver input files must be gathered together in one directory and solver executables in another. The absolute paths to the directories containing input files and executables must be given in paths.py as follows:

\[
\text{SPECFEM\_DATA} = /\text{path/to/spcelfem/input/files} \\
\text{SPECFEM\_BIN} = /\text{path/to/specfem/executable/files}
\]

6.5.4 Writing Custom Solver Interfaces

Besides SPECFEM2D, SPECFEM3D, and SPECFEM3D_GLOBE, SeisFlows can interface with other solvers. Users unaffiliated with the main SeisFlows developers have succeeded in interfacing with, for example, their own finite difference solvers.

Integration of the solver with the other workflow components can be challenging. Here we try to give an idea of the issues involved from both a developer and a user standpoint.
- Solver computations account for most of the cost of an inversion. As a result, the solver must be written in an efficient compiled language, and wrappers must be written to integrate the compiled code with other software components.

- There is currently no mechanism for automatically compiling executables. Users must prepare their own solver input files and then follow the compilation procedure in the solver documentation.

- *SeisFlows* uses two input files, `paths.py` and `parameters.py`. Problems could arise if parameters from *SeisFlows* input files conflict with parameters from solver input file. Users must make sure that there are no conflicts between *SeisFlows* parameters and solver parameters.

- In the solver routines, it’s natural to represent velocity models as dictionaries, with different keys corresponding to different material parameters. In the optimization routines, it’s natural to represent velocity models as vectors. To convert back and forth between these two representations, a pair of utility functions—*split* and *merge*—are included in solver.base.

### 6.5.5 System Configuration

*SeisFlows* can run on SLURM, PBS, and LSF clusters, as well as, for very small problems, laptops or desktops. A list of available system interface classes follows. By hiding environment details behind a python interface layer, these classes provide a consistent command set across different computing environments.

**PBS_SM** For small inversions on PBS clusters. All resources are allocated at the beginning and all simulations are run within a single job. Requires that individual wavefield simulations run each on a single core, making this option suitable for small 2D inversions only.

**PBS_LG** For large inversions on PBS clusters. The work of the inversion is divided between multiple jobs that are coordinated by a single long-running master
job. Resources are allocated on a per simulation basis. Suitable for small to medium 3D inversions in which individual wavefield simulation span several or more nodes.

**SLURM_SM** For small inversions on SLURM clusters. All resources are allocated at the beginning and all simulations are run within a single job. Requires that each individual wavefield simulation runs only a single core, making this option suitable for small 2D inversions only.

**SLURM_MD** For small to moderate-sized inversions on SLURM clusters. All resources are allocated at the beginning and all simulations are run within a single job. Individual wavefield simulations can span more than one core, but not more than one node. Suitable mainly for 2D inversions, although some small 3D inversion might be possible.

**SLURM_LG** For large inversions on SLURM clusters. The work of the inversion is divided between multiple jobs that are coordinated by a single long-running master job. Resources are allocated on a per simulation basis. Suitable for 3D inversions in which individual wavefield simulation span several or more nodes.

**SLURM_XL** For large inversions on SLURM clusters. In addition to the features of SLURM_LG, provides fault tolerance. Tasks that end in failure or timeout are automatically resubmitted. For this reason, can be dangerous to use on code that is not well tested.

**SERIAL** Tasks that are normally carried out in parallel are instead carried out one at a time. Useful for debugging, but not much else.

**MULTITHREADED** On desktops or laptops with multiple cores, allows embarrassingly parallel tasks to be carried out several at a time, rather than one at a time. Can be used to run small 2D inversions on a laptop or desktop.

**MPI** Similar in functionality to MULTITHREADED, except uses MPI processes rather than multithreading for parallelism. Requires Python module mpi4py.

**LSF_SM** Same as SLURM_SM and PBS_SM, except for LSF clusters.
**LSF\_LG** Same as SLURM\_LG and PBS\_LG, except for LSF clusters.

**PBS\_TORQUE\_SM** Same as PBS\_SM, except uses `pbsdsh` rather than `mpi4py` under the hood.

**TIGER\_SM** Slightly specialized version of SLURM\_SM made available for Princeton users.

**TIGER\_MD** Slightly specialized version of SLURM\_MD made available for Princeton users.

**TIGER\_LG** Slightly specialized version of SLURM\_LG made available for Princeton users.

**TIGER\_MD\_GPU** Highly specialized version of SLURM\_MD made available for Princeton GPU users. Provided by Etienne Bachmann. Not recently tested and not likely to work right out of the box.

### 6.5.6 Writing Custom System Interfaces

If your needs are more specialized, please view `seisflows.system` source code to get a sense for how to write your own custom system interfaces. In our experience, system interfaces require no more than a few hundred lines of code, so writing your own is generally possible once you are familiar with the SeisFlows framework and your own cluster environment.

To make SeisFlows work across different environments, our approach is to wrap system commands with a thin Python layer. To handle job submission, for example, we wrap the PBS command `qsub` and the SLURM command `sbatch` with a python utility called `system.submit`. The result is a consistent python interface across different clusters.

Filesystem settings can be adjusted by modifying values in the `PATH` dictionary, which is populated from `paths.py`. Output files and temporary files, by default, are written to the working directory. If a value for `PATH.SCRATCH` is supplied, temporary
files are written there instead. If each compute node has its own local filesystem, a value for `PATH.LOCAL` can be supplied so that temporary files required only for a local process need not be written to the global filesystem.

As the size of an inversion grows, scalability and fault tolerance become increasingly important. If a single forward simulation spans more than one node, users must select `pbs lg` or `slurm lg` system configurations in `parameters.py`. If a forward simulation fits onto a single node, users should select `pbs sm` or `slurm sm` instead.

In SeisFlows, the overall approach to solving system interface problems is to use lightweight Python wrappers. For complex cluster configurations, heavier-weight solutions may be required. Users are referred to SAGA or Pegasus projects for ideas.

### 6.5.7 Developer Reference

To allow classes to work with one another, each must conform to an established interface. This means certain classes must implement certain methods, with specified input and output. Required methods include

- **setup** methods are generic methods, called from the `main` workflow script and meant to provide users the flexibility to perform any required setup tasks.

- **check** methods are the default mechanism for parameter declaration and checking and are called just once, prior to a job being submitted through the scheduler.

Besides required methods, classes may include any number of private methods or utility functions.

### 6.5.8 Parameter Files

`parameters.py` contains a list of parameter names and values. Prior to a job being submitted, parameters are checked so that errors can be detected without loss of queue time or wall time. Parameters are stored in a dictionary that is accessible from anywhere in the Python code. By convention, all parameter names must be upper
case. Parameter values can be floats, integers, strings or any other Python data type. Parameters can be listed in any order.

paths.py contains a list of path names and values. Prior to a job being submitted, paths are checked so that errors can be detected without loss of queue time or wall time. Paths are stored in a dictionary that is accessible from anywhere in the Python code. By convention, all names must be upper case, and all values must be absolute paths. Paths can be listed in any order.
Chapter 7

Conclusions

In this thesis we employed a “brute force” approach to identify best practices. From hundreds of inversions with different target models, nonlinear optimization algorithms, preconditioners, regularization methods, objective functions, and material parameterizations, both expected and unexpected patterns emerged.

Among all the acoustic brute force results, the nonlinear optimization results were perhaps the least surprising. Not only did L-BFGS provide faster convergence than NLCG in all test cases, the relative performance of the two algorithms and the contribution from the line search to total computational cost agreed with experience from the general numerical optimization literature. Results of the regularization experiments, in contrast, were sometimes surprising. Even though the methods considered were to one degree or another classical, new details emerged from analyzing them in the waveform inversion context, and performance sometimes differed significantly from one waveform inversion test case to another.

If the acoustic inversion results were sometimes unexpected, the elastic inversion results were even more so. Conventional scattering analysis, which has been used for decades as a device for investigating material parameterizations, was found to provide a poor substitute for actual inversion experiments. When measured in terms of
convergence rate directly, the relative performance of parameterizations in some cases turns out to be quite different than analytical scattering devices predict. The relative performance of different material parameterizations, moreover, varies depending on the objective function and the target model in ways that scattering analysis fails to anticipate.

Although our main use of the brute force approach was to identify best practices, the experiments also suggest avenues for further research. While comparisons between L-BFGS and NLCG in the acoustic context were largely unsurprisingly, comparisons between quasi-Newton and truncated Newton methods in the elastic context point to a lack of agreement in the current literature. Further work is required to sort out what benefits truncated Newton methods provide in terms of efficiency and robustness and to reconcile these results with the general applied mathematics literature. At the same time, regularization experiments highlight the need for additional work on $L_1$ penalty functions. Considering the wide variety of $L_1$ nondifferentiability workarounds proposed in the numerical optimization literature that have yet to be tested in waveform inversion, it might be worthwhile to the brute force approach to bear on this issue. Finally, the experimental method developed for comparing isotropic elastic material parameterizations could be extended to anisotropic elastic parameterizations. Taking advantage of computational resources far greater than those available to the earliest waveform inversion practitioners, a comparison of anisotropic parameterizations could lead to physical intuition as to which parameters map to which aspects of the data, numerical intuition including the possibility of general nonlinear preconditioners not tied to familiar physical moduli, and ultimately to improved the understanding of the shallow and deep subsurface through more robust and efficient recovery of anisotropic structure.
Appendix A

Implementation Details

A.1 Limited-memory BFGS algorithm

Main algorithm
Given an initial model $m_0$, objective function $f$, diagonal scaling $D$, memory value $l$, and stopping threshold $\delta > 0$, the L-BFGS algorithm is as follows:

1. Evaluate $f_0 = f(m_0), \ g_0 = \nabla f(m_0)$.

2. Set $p_0 = -g_0, \ k = 0$.

3. If $k > 0$, compute $p_k$ from recursion, below.

4. Compute $\alpha_k$ by line search and set $m_{k+1} = m_k + \alpha_k p_k$.

5. Evaluate $f_{k+1} = f(m_{k+1}), \ g_{k+1} = \nabla f(m_{k+1})$.

6. Set $s_k = m_{k+1} - m_k, \ y_k = g_{k+1} - g_k, \ k = k + 1$.

7. Repeat (3–6), stopping immediately when $g_{k+1}^T g_{k+1} < \delta$.

Recursion

1. Set $q = g_k, \ i = k - 1, \ j = \min(k, l)$.  

150
2. Perform \( j \) times: 
\[
\lambda_i = s_i^T q, \quad q = q - \lambda_i y_i, \quad i = i - 1.
\]

3. Set 
\[
\gamma = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}, \quad r = \gamma D q, \quad i = k - j.
\]

4. Perform \( j \) times: 
\[
\mu = s_i^T r, \quad r = r + s_i (\lambda_i - \mu), \quad i = i + 1.
\]

5. End with result \( p_k = -r \).

Remarks

L-BFGS’s relatively modest memory usage rests on the fact that if \( k \) is the current iteration number and \( l \) is the memory value, then vector pairs prior to \( \{s_{k-l}, y_{k-l}\} \) are no longer needed and can be removed from storage.

The scaling factor \( \gamma \), which accounts for differences between the true Hessian and the approximation thereto, is essential to the good performance of the algorithm. Of several choices proposed by Liu and Nocedal (1989), the expression for \( \gamma \) given above (step 3 of recursion) has been found to provide the best results.

A.2 Preconditioned nonlinear conjugate gradient method

Given an initial model \( m_0 \), objective function \( f \), preconditioner \( P \), and stopping threshold \( \delta > 0 \), the preconditioned NLCG method is as follows:

1. Evaluate \( f_0 = f(m_0), \quad g_0 = \nabla f(m_0) \).

2. Solve \( Py_0 = g_0 \).

3. Set \( p_0 = -y_0, \quad k = 0 \).

4. Compute \( \alpha_k \) by line search and set \( m_{k+1} = m_k + \alpha_k p_k \).

5. Evaluate \( f_{k+1} = f(m_{k+1}), \quad g_{k+1} = \nabla f(m_{k+1}) \).
6. Solve $Py_{k+1} = g_{k+1}$.

7. Set $\beta_{k+1} = \frac{g_{k+1}^T(y_{k+1}-y_k)}{g_k^T y_k}$, $p_{k+1} = -y_{k+1} + \beta_{k+1} p_k$, $k = k + 1$.

8. Repeat (5–8) until $g_{k+1}^T g_{k+1} < \delta$.

The precise form of the algorithm above is due to Polak and Ribiére (1969). Besides this one, a number of other variants exist. One due to Fletcher and Reeves (1964) may be slightly worse, and another due to Gilbert and Nocedal (1992) may be slightly better. Since savings from the use of L-BFGS over NLCG are in general much larger than savings from the use of one NLCG algorithm over another, we have not investigated which of these variants performs best in the waveform inversion context.

### A.3 Restart conditions

Even with robust regularization and multiscale methods, numerical difficulties in waveform inversion are neither unexpected nor uncommon. Below, we describe two restart conditions that can be effective in addressing such problems.

**Angle restart condition**

In section 4.4, we discussed the possibility that a search direction is not actually a descent direction. For poorly-conditioned problems, it may be beneficial to impose even stricter restart conditions than $p_k^T g_k > 0$.

One possibility for such a condition is

\[
\frac{p_k^T g_k}{\sqrt{p_k^T p_k} \sqrt{g_k^T g_k}} > \tau,
\]

where $-1 < \tau < 0$ is some user-supplied parameter. Values of $\tau$ of about $-0.02$, we find, are usually effective.
The above restart condition can be reformulated in terms of the angle $\theta$ between the gradient and the search direction using the relation $\theta = \arccos(\tau)$. For example, the above condition with $\tau = -0.087$ is equivalent to requiring that $\theta > 95$.

**Powell restart condition**

In section 4.4, we mentioned the tendency of NLCG search directions to lose conjugacy. As described by Powell (1977), the restart condition

$$\frac{g_k^T g_{k+1}}{g_k^T g_k} > \tau,$$

where $\tau > 0$ is some user-supplied parameter, provides an effective workaround. A common choice for $\tau$ is 0.2.

Since conjugacy is expected of NLCG search directions but not L-BFGS search directions, Powell restart conditions can be used in combination with with the former but not the latter. While performance after Powell restarts can be poor in the short time, the possibility of substantially better long term performance makes the procedure worthwhile.

### A.4 Line search termination conditions

Given an objective function $f$, model $m$ and search direction $p$, and letting $\phi(\alpha) = f(m + \alpha p)$, the work of the line search is to find a step length $\alpha$ such that the updated model $m + \alpha p$ meets the termination conditions

$$\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'(0),$$

$$\phi'(\alpha) \geq c_2 \phi'(0),$$

where $c_1 > 0$ and $0 < c_2 < 1$ are user-supplied numerical parameters. The first condition above is called the *Armijo condition* or *sufficient decrease condition*. The
second is called the *curvature condition*. Both together are known as the *Wolfe conditions*.

It had been found that L-BFGS is best implemented with a very loose line search. Following Liu and Nocedal (1989) and many other studies, a common choice of numerical parameters is $c_1 = 10^{-4}$ and $c_2 = 0.9$. NLCG is often implemented with a somewhat stricter line search, along the lines of $c_1 = 10^{-4}$ and $c_2 = 0.1$. With the NLCG bracketing line search described in section 5.1, however, we find it more cost effective to use $c_1 = 10^{-4}$ and $c_2 = 0.9$ to reduce the number of gradient evaluations, the bracketing and interpolation requirements being enough, it seems, to ensure a sufficiently accurate step length.
Appendix B

Reproducibility

All the acoustic inversion experiments from chapter 2 can be reproduced using the open source SeisFlows package, which resides at github.com/PrincetonUniversity/seisflows. To avoid overcrowding the main package, many of the elastic and anisotropic strategies described in chapters 3–5 are implemented under a separate repository located at bitbucket.com/rmodrak/seisflows-research.

Because of their large size, the model and configuration files used in the acoustic test cases are not hosted on a publicly-accessible server. Instead, the models are available as a set of binary files for any Princeton account holder. For readers without such an account, the checkerboard test case used to illustrate regularization and preconditioning concepts in chapter 2 is available as a fully functional example. This section provides step-by-step instructions for installing SeisFlows, running the checkerboard inversion and visualizing the results.
B.1 Checkerboard example

B.1.1 Download SeisFlows

To run SeisFlows you’ll need a Unix system with Python 2.7, Numpy, Scipy, and standard Unix utilities. After these prerequisites are in place, from the command line type:

```
mkdir ~/.packages
cd ~/.packages

git clone https://github.com/PrincetonUniversity/seisflows.git
```

If you prefer a location other than `/packages`, modify the commands above and below accordingly.

B.1.2 Set environment variables

Add the following lines to `.bash_profile` (modify accordingly, if you are using a shell other than bash):

```
export PATH=$PATH:~/packages/seisflows/scripts
export PYTHONPATH=~/packages/seisflows
```

Don’t forget to update any open shells:

```
source ~/.bash_profile
```

B.1.3 Run “system” test

Run the following test to make sure everything is working:

```
cd ~/packages/seisflows/tests/test_system
./clean.py; ./run.py
```

If a “hello” message is displayed, the test was successful.
B.1.4 Run nonlinear optimization test

Run the following test to make sure everything is working:

```
    cd ~/packages/seisflows/tests/test_optimize
    ./clean.py; ./run.py
```

If the optimization problem is solved in 50 iterations or fewer, the test was successful.

B.1.5 Configure and compile SPECFEM2D

First, download SPECFEM2D from GitHub:

```
    cd ~/packages
    git clone --recursive --branch devel https://github.com/geodynamics/S
    cd SPECFEM2D-d745c542
    git checkout d745c542
```

For now, it is important to work with the exact version specified above (d745c542). This is necessary because, unlike SPECFEM3D, SPECFEM2D development is largely unfunded and sometimes a bit haphazard, with frequent interface changes.

Next, configure and compile SPECFEM2D using ifort (preferred) or gfortran:

```
    cd ~/packages/SPECFEM2D-d745c542
    ./configure FC=ifort
    make all
```

(Since ‘make’ by itself does not compile all the required utilities, be sure to remember to type ‘make all’.) For troubleshooting any compilation issues, please view the SPECFEM2D manual and GitHub issues page.
B.1.6 Set up checkerboard test

Download the starting model and other input files required for the waveform inversion checkerboard test. For simplicity, let’s assume the checkerboard working directory will be placed in /tests (if you prefer a different location, then modify the following commands accordingly):

```
mkdir ~/tests/
cd ~/tests/
wget --recursive --no-parent --no-host-directories
    --cut-dirs=2 --reject "index.html*" \
    http://tigress-web.princeton.edu/~rmodrak/2dAcoustic/
```

A directory /tests/checkers is now being created. Among other files, parameters.py and paths.py are being downloaded.

After the download completes, make sure that all paths specified in paths.py are correct. For example, if you compiled SPECFEM2D somewhere other than /packages/SPECFEM2D-d745c542, you will need to modify the SPECFEM2D_BIN entry accordingly.

Next, take a minute to view the parameters.py file and note the close similarity between the first set of parameters and the directory structure of the SeisFlows source code.

B.1.7 Run checkerboard test in serial

To run the checkerboard test type:

```
sfclean ; sfrun
```

within /tests/checkers.
For now, the inversion will run only a single event on only a single processor. Once we verify that everything is working correctly in this case, we can move on to multiple events and multiple processors by modifying parameters.py.

### B.1.8 Rueen checkerboard test in parallel

On a laptop or desktop with multiple cores, the work of an inversion can be carried out in parallel. To run the checkerboard example in parallel over events (that is, with multiple event simulations running at the same time on different cores), make the following changes to parameters.py:

- to invert all available events instead of just one event, change NTASK from 1 to 25
- change SYSTEM from serial to multithreaded
- add a parameter NPROCMAX and set it to the number of cores available on your machine.

Besides running in parallel over events, the work of an individual event simulation can be parallelized over model regions. See the SPECFEM3D user manual for more information. Both parallelization over events and over model regions can be used at the same time under SeisFlows. The current example, however, illustrates only event parallelism.

Besides serial and multithreaded settings for running SeisFlows on laptops and desktops, there are also PBS, SLURM, and LSF options for running on clusters.

### B.2 Visualization

Visualization requires software such as Pylab, Matlab, or Paraview. With any such software, one approach for plotting SPECFEM2D models or kernels is to inter-
polate from the unstructured numerical mesh on which the model parameters are defined to a uniform rectangular grid. The Pylab script 'plot2d ¡http://tigress-web.princeton.edu/ rmodrak/visualize/plot2d¡:' illustrates this approach. Another method is to compute a Delaunay triangulation and plot the model or kernel over the unstructured mesh itself.
Appendix C

Supplemental Figures: Acoustic Inversion

1. Performance of L-BFGS with different memory values
2. Performance of NLCG with $P_1$ preconditioners
3. Performance of L-BFGS with $P_1$ initial scalings
4. Performance of L-BFGS with $P_3$ initial scalings
5. Brute force regularization parameter selection: effect of varying Gaussian smoothing parameter
6. Brute force regularization parameter selection: effect of varying number of basis functions
7. Brute force regularization parameter selection: effect of varying Tikhonov regularization weight
8. Brute force regularization parameter selection: effect of varying TV regularization weight
9. Effect of restarting optimization algorithm at multiscale transitions
10. Comparison of water layer masking strategies
11. Role of regularization in suppressing nonuniqueness
Figure C.1: Performance of L-BFGS with different memory values
Figure C.2: Performance of NLCG with $P_1$ preconditioners
Figure C.3: Performance of L–BFGS with $P_1$ initial scalings
Figure C.4: Performance of L–BFGS with $P_3$ initial scalings
Figure C.5: Brute force parameter selection experiments: effect of varying Gaussian smoothing parameter
Figure C.6: Brute force parameter selection experiments: effect of varying the number of Gaussian basis functions used to represent the model. The ratio $f$ is the number of basis functions divided by the number of points in the numerical grid. For a given test case, the former is varied from one experiment to another and the latter is constant.
Figure C.7: Brute force parameter selection experiments: effect of varying Tikhonov regularization weight
Figure C.8: Brute force parameter selection experiments: effect of varying total variation regularization parameter
Figure C.9: Effect of restarting optimization algorithm at multiscale transitions

Figure C.10: Strategies for masking a water layer
Figure C.11: Role of regularization in suppressing nonuniqueness illustrated through a checkerboard example. Each panel above shows the error in the inversion result after 25 updates from a homogeneous starting model, with the standard deviation $\sigma$ of the Gaussian “regularization by convolution” kernel varied from one panel to another.
Bibliography


174


