VELOCITY MODEL BUILDING USING SHAPE OPTIMIZATION APPLIED TO LEVEL SETS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF GEOPHYSICS
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Taylor Dahlke
May 2019

Printed as Stanford Exploration Project No. 175
by permission of the author

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Biondo L. Biondi) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Gregory Beroza)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Robert G. Clapp)

Approved for the Stanford University Committee on Graduate Studies

Abstract

As oil and gas extraction becomes more advanced, exploration becomes increasingly focused on imaging near or under complex salt geology, which necessitates detailed velocity models with sharp interfaces. Current state-of-the-art practices include the use of Full-Waveform Inversion (FWI). However, this type of approach can result in salt body models that lack the sharp interfaces characteristic of this type of geobody. This is typically due to the computational expense of using increasingly high frequencies in the data. These interfaces can be elegantly tracked as sharp boundaries using the level sets of an implicit surface. Used in conjunction with shape optimization, one can invert for salt boundaries that fit recorded data in a FWI style objective function that is parameterized in terms of both the implicit surface and a background velocity model. While this addition of the implicit surface requires more model parameters, radial basis functions can be used to create a sparse parameterization of it, which can hasten convergence of the inversion. The implicit surface also allows for embedding information about the certainty of different salt boundary regions by means of its initialization. This information allows for intelligent guidance of the inversion based on interpreter input, which can help the inversion avoid local minima. The result of testing this inversion workflow on a 3D Gulf of Mexico dataset shows that it can be a useful tool for refining salt models, as the seismic images produced from the final model shows clearer and more consistent features below the updated salt area.

Preface

The electronic version of this report[1] makes the included programs and applications available to the reader. The markings **ER**, **CR**, and **NR** are promises by the author about the reproducibility of each figure result. Reproducibility is a way of organizing computational research that allows both the author and the reader of a publication to verify the reported results. Reproducibility facilitates the transfer of knowledge within SEP and between SEP and its sponsors.

ER denotes Easily Reproducible and are the results of processing described in the paper. The author claims that you can reproduce such a figure from the programs, parameters, and makefiles included in the electronic document. The data must either be included in the electronic distribution, be easily available to all researchers (e.g., SEG-EAGE data sets), or be available in the SEP data library[2]. We assume you have a UNIX workstation with Fortran, Fortran90, C, X-Windows system and the software downloadable from our website (SEP makerules, SEPScons, SEPlib, and the SEP latex package), or other free software such as SU. Before the publication of the electronic document, someone other than the author tests the author's claim by destroying and rebuilding all ER figures. Some ER figures may not be reproducible by outsiders because they depend on data sets that are too large to distribute, or data that we do not have permission to redistribute but are in the SEP data library, or that the rules depend on commercial packages such as Matlab or Mathematica.

CR denotes Conditional Reproducibility. The author certifies that the commands are in place to reproduce the figure if certain resources are available. The primary reasons for the CR designation is that the processing requires 20 minutes

[1]http://sepwww.stanford.edu/public/docs/sep165
[2]http://sepwww.stanford.edu/public/docs/sepdatalib/toc_html/

or more.

NR denotes Non-Reproducible figures. SEP discourages authors from flagging their figures as NR except for figures that are used solely for motivation, comparison, or illustration of the theory, such as: artist drawings, scannings, or figures taken from SEP reports not by the authors or from non-SEP publications.

Our testing is currently limited to LINUX 2.7 (using the Intel Fortran90 compiler) and the SEPlib-7.0.5 distribution, but the code should be portable to other architectures. Reader's suggestions are welcome. For more information on reproducing SEP's electronic documents, please visit *http://sepwww.stanford.edu/research/redoc/*.

Acknowledgments

A thesis has a single author, but it is truly a collaboration of many individuals, without whom the final result would never be realized. I am grateful to so many individuals who have supported and helped me finally achieve this dream.

When I began at Stanford, I was lucky enough to find myself among other peers who have made this experience an educational and enjoyable one. Chief among them was my cohort in geophysics including Alex Hakso, Badr Rumiah, Chris Castillo, Humberto Samuel Arevalo Lopez, Fatemeh Rassouli, Lei Jin, Priyanka Dutta, and many more. The gatherings we organized over the years are fond memories, and raised my spirit many times. More recently, I am grateful to Leighton Watson and Hiroshi Mendoza for their friendship, practical help, many conversations, and shared adventures.

The Stanford Exploration Project (SEP) is a group of people who are set apart by their intelligence, work ethic, traditions, and diversity of background. I've been fortunate to have made so many friends, and have received help from so many talented people. A number of them graduated before me including Sjoerd de Ridder, Adam Halpert, Elita Li, Mandy Wong, Xukai Shen, Zhang Yang, Chris Leader, Ali Almomin, Musa Maharramov, Yi Shen, Ohad Barak, Gustavo Alves, Jason Chang, and Eileen Martin. I am grateful to Mandy and Victor for letting me live with them in Houston several times, and demonstrating so well what it looks like to be hospitable. Jason Chang was my officemate for more than five years, and his low-key and cheerful demeanor has been most welcome. Ohad Barak taught me a tremendous deal about Linux systems, and his kind guidance has often made me feel at ease. My SEP peers Eileen Martin and Huy Le have always made themselves accessible to my lengthy questions, even when they were at their busiest. I've have enjoyed the friendship of all of them.

I have also been the benefactor of learning from the students who began after me, including Guillaume Barnier, Ettore Biondi, Yinbin Ma, Alejandro Cabrales, Joseph Jennings, Fantine Hout, Rahul Sarkar, Stuart Farris, Siyuan Yuan, Miguel Ferrer Avila, Rustam Akhmadiev, and Milad Bader. They each have been generous in freely offering their knowledge to me over countless occasions. It does sadden me to be leaving such a talented group of people as I transition on to new things.

I thank Shell Exploration and Production Company for providing me with the 3D field dataset used in chapter five of this thesis. Over the last few years I've had the privilege of working with a number of talented people at Shell. Bonnie Jones has been especially helpful with the 3D field data, and has answered many last minute queries for me with much patience. Mauricio Araya Polo and Detlef Hohl gave me the opportunity to diversify my skill set and work on cutting edge research with neural networks, and I am grateful for their instruction and support. I also want to thank Jan Douma for his help with the 3D dataset. I would further like to thank each of them for their hospitality while I was living in Houston.

I leave a wealth of knowledge behind as I say farewell to Stew Levin, Shuki Ronen, Jon Claerbout, Dennis Michael, Robert Clapp, Gregory Beroza, and my advisor Biondo Biondi. Stew Levin offers kindness and knowledge to all who encounter him, including myself. Shuki Ronen has been instrumental in helping me with many signal processing problems. Robert Clapp has been a tremendous resource for me over the years, and I have witnessed his unstoppable intelligence and aptitude save the day for myself, other students, and the entire group on many occasions. Jon Claerbout has always been an example to me of lifetime learning and adaptation. Dennis Michael has been a tremendous resource to me regarding computation throughout my time at Stanford. Gregory Beroza has offered thoughtful insights into my work many times in regards to both my primary and secondary projects at Stanford. Finally, I am grateful to Biondo Biondi for his patience teaching me many topics over the years, and for his insistence on objectivity both in the rigors of geophysics, as well as in the way he conducts himself and the group.

I have been lucky enough to have the support of steadfast friends outside of the Stanford community, including David and Jonah Turnquist, Jason Smith, Joseph Cooney, Nick Shelton, Johnson Kang, Nathan Tsoi, and many others. They have sharpened me, challenged me, and supported me in this endeavor in more ways than they know. I am grateful to Brad and Lynn Smith for providing a workspace for me to finish the final six months of my thesis, which was invaluable. The Mundy family and the Lahl family have been encouraging and supportive at many stages of this journey. I am blessed to have them in my life.

My family has been a part of my educational story long before Stanford was on the horizon. I am forever grateful to my parents for making the sacrifices to homeschool me, despite the challenges and opportunity costs to themselves. They have pushed me to do better and be persistent in situations where on my own strength I may well have backed down. They planted and watered the seeds of my curiosity from the earliest age, and I would never have been able to start this degree, let alone finish it, without their uncompromising encouragement and support. I am grateful to my brother Zachary and sister Annie for being my first classmates and friends, for always challenging me to do my best, and for inspiring me in the many times they have outdone me. I continue to learn from them as we journey through life together.

When I began at Stanford, I thought that the best thing that could happen to me more than six years later would be finally finishing my degree. How wrong I truly was! Along the way I was lucky enough to meet a woman who has changed my life in more ways than I can explain. During the hardest parts of this journey, when it felt like I was stalling out going up a steep hill, Emily has always been the first to jump out and push. I am so lucky to call her my wife, and to have her beside me as we finally make it to the top.

"*For I know the plans I have for you, declares the Lord, plans to prosper you and not to harm you, plans to give you hope and a future. Then you will call on me and come and pray to me, and I will listen to you. You will seek me and find me when you seek me with all your heart.*"

– – – Jeremiah 29:11-13 (NIV)

Contents

List of Tables

List of Figures

Chapter 1

Introduction

PROBLEM OVERVIEW

Why is salt hard to model?

Oil producing regions like the Gulf of Mexico are known to have geologically complex salt body formations. These formations can act as seals, trapping hydrocarbons underneath, which makes them frequent targets of seismic imaging projects. However, salt bodies have two important properties that make them difficult to image. First, the P-wave velocity of salt is high, and contrasts sharply with that of surrounding sediment layers, making them very reflective to seismic energy. Second, salt bodies can adopt a variety of complex shapes and geometries. Because of these factors, the formations themselves act as lenses which focus or disperse seismic energy (Leveille et al., 2011; Etgen et al., 2016; Barnier and Biondi, 2015). Because salt bodies often have steep dipping boundaries, useful reflection energy may be directed along raypaths that reach the surface far outside of the acquisition geometry extent (see Figure 1.1(b)). With less complex imaging subjects (Figure 1.1(a)) this is not so often the case. Lack of data capture can subsequently impact the imaging results by creating 'blind' spots near the base and flanks of the salt. Figure 1.2(a) demonstrates how flat portions of the salt boundary have good illumination, while the steeply dipping flank (Figure 1.2(b)) has poor illumination, making picking the salt boundary more difficult.

However, even when reflected energy is sufficiently recorded, strong reflectivity

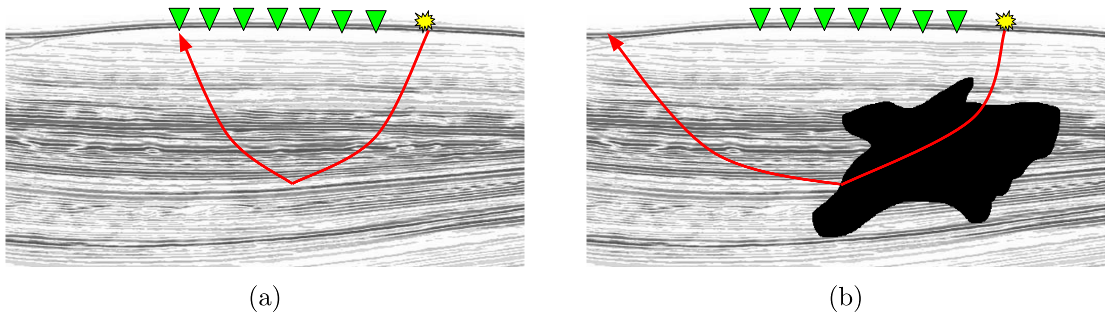


Figure 1.1: a) Ray path from flat reflector. b) Ray path from salt reflector. Note that after passing through salt (black) the raypath in 1.1(b) reaches the surface outside of the acquisition zone. Receivers (green) and source (yellow) are shown on the surface. [**NR**] $\boxed{\text{chapter1/. ray-path-no-salt,ray-path-with-salt}}$

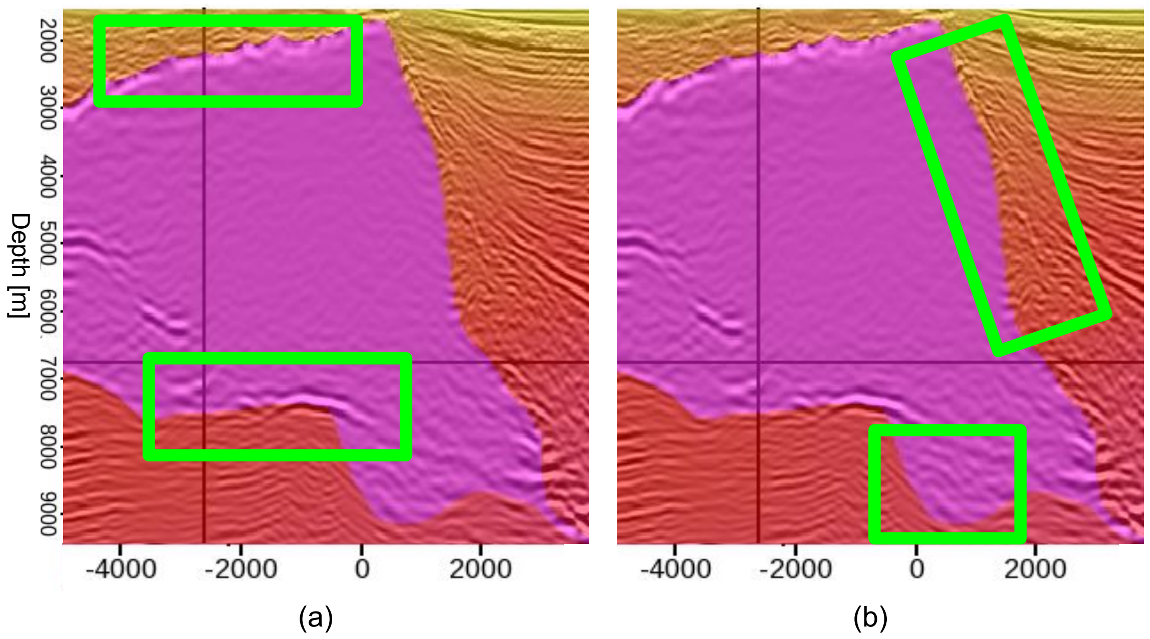


Figure 1.2: Salt model (pink) overlain on seismic image. Green boxes in a) highlight areas of good illumination, while boxes in b) highlight areas of poorer illumination. [**NR**] $\boxed{\text{chapter1/. salt-picking-examp}}$

and lensing effects can amplify small errors in the salt model and negatively impact the resulting image. The boundaries of salt are typically improved in an iterative workflow (interpret salt from image, update salt model, then re-migrate image as in Figure 1.3). Subsequently, small initial errors may be compounded as velocity model development proceeds. Furthermore, these methods are time-consuming and tedious since expert input is necessary for either the actual picking, or the oversight and correction when picking is semi-automated. Each iteration of this approach can take days or weeks while coordinating entire teams of people. A robust method for further automating the salt model building procedure would greatly alleviate this bottleneck.

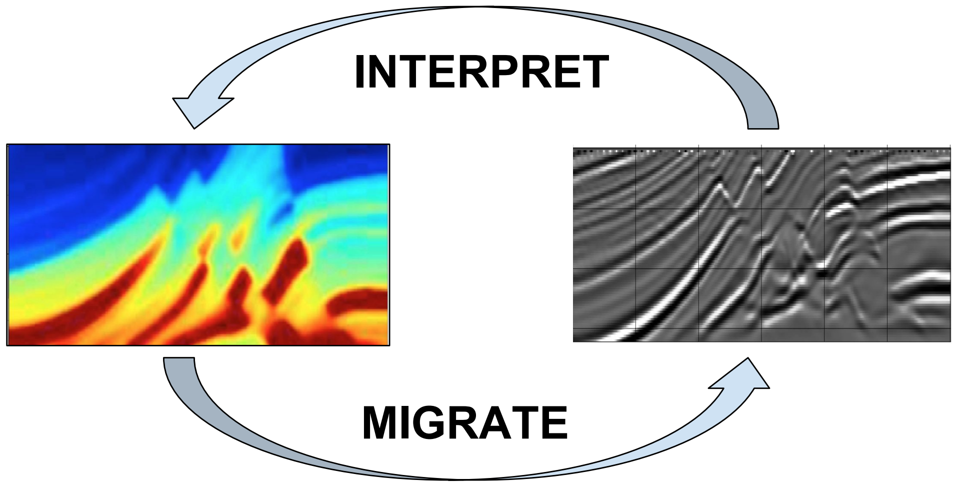


Figure 1.3: Basic velocity model building workflow concept. [**NR**] chapter1/. vel-building-workflow

Approaches like full-waveform inversion (FWI) are intended to avoid the repeated manual interpretation necessary in the approach just described, but can be less than effective for a variety of reasons. Sometimes high frequencies in the data are of poor quality, causing difficulty reconstructing sharp velocity models. When high-frequencies are present with sufficient quality, multi-scale inversion workflows are typically used, which build a model starting from low and working up to high frequencies Bunks et al. (1995). However, to increase the sharpness of the model requires higher and higher frequencies, which become increasingly expensive to compute wave propagations for. To alleviate this expense, multi-scale inversion is sometimes done using wider frequency band increments, or even using a full band of frequencies from the beginning.

Unfortunately, using high frequencies too early in the inversion runs a higher risk of converging to a local minima. This is because the FWI objective function is based on the data residual, and the size of the 'well of convergence' around the true model is based on the wavelengths in the data (see Figure 1.4(e), pink). The 'well of convergence' is the model-space region surrounding the true model where the objective function is convex. When a starting model is within this region, the objective function gradient will eventually direct convergence to the true model. When a shift in the modeled data is greater than a half wavelength from the true data, the objective function gradient will direct updating to a local minima, pushing the data to align with a wave cycle other than the one representing the true model (i.e. skipping a wave cycle). With low frequencies the wavelength is longer and the well of convergence is bigger (Figure 1.4(e)), and thus greater model error can be tolerated Virieux and Operto (2009). However, even using advanced FWI tools with starting models and frequencies that thoughtfully consider cycle-skipping, the final results typically lack sharp resolution around the salt edge (compare Figures 1.5(a), 1.5(b)), which can impact the clarity of seismic imaging near the salt.

One approach that is used to create sharp boundaries in velocity models is Total-Variation (TV) regularization, which makes use of the L1 norm to regularize the model parameters during the inversion (Maharramov and Levin (2015), Esser et al. (2018)). This creates a regularized version of the FWI objective function that includes an additional term, $\lambda|Am|_1$, where A is a Laplacian operator that takes the spatial derivative of the model space. There are several challenges with this approach. The first is that a parameter λ must be chosen which appropriately balances the regularization term with the data residual term. This is can be difficult to find, and may even need to change with iteration for the best inversion result. The second problem is that taking the derivative of this new term is difficult. This is because the L1 norm is not a differentiable function at zero. This means that an optimization method must be chosen for a non-smooth problem, such as the subgradient method (Maharramov, 2016), which can be much slower than Newton's method on a smooth objective function.

Why are level sets the answer?

A key tool for addressing this problem is the concept of the level set. A level set is a contour of a higher dimensional surface used to track a discrete boundary. In the salt

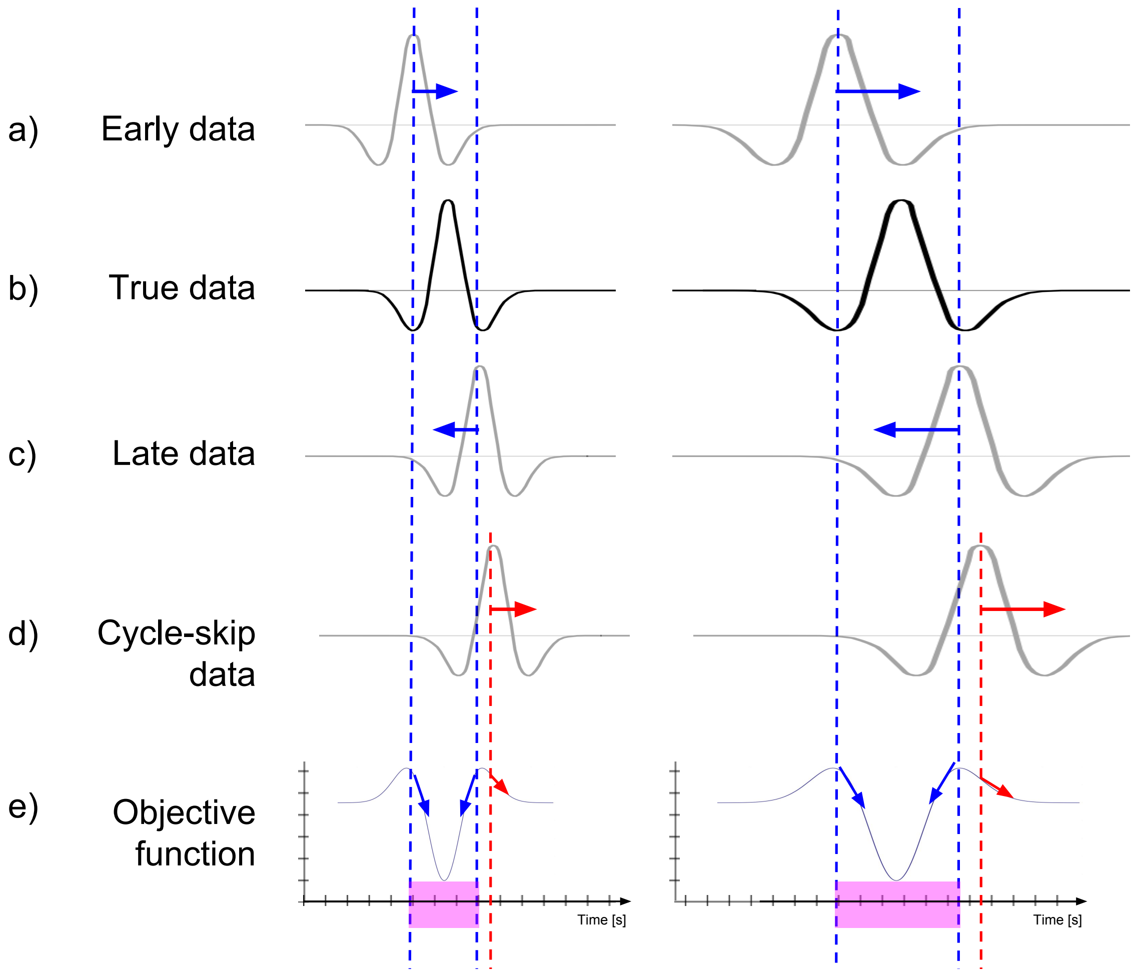
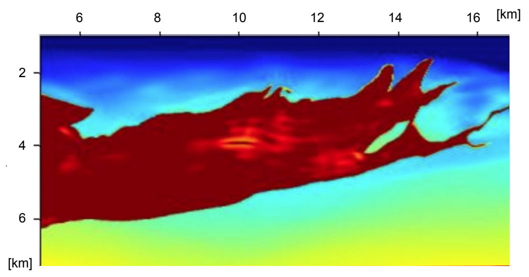
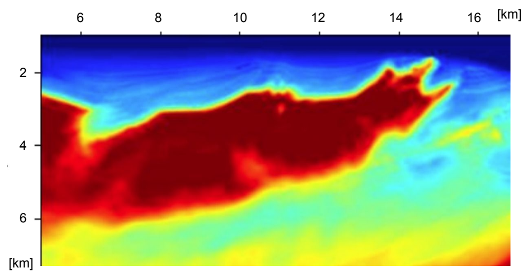


Figure 1.4: Left column shows higher frequency data while right column shows lower frequency data. True data (b) and shifted traces (a,c), with the objective function (e) showing the data residual norm measured at all shift positions. When a trace is shifted far enough from the true trace (d), the gradient of the objective function directs updating to a local minima, pushing the trace *further* from the true position (red arrows), and cycle skipping occurs. Blue arrows indicate convergence towards the true solution when the modeled trace shift (and thus model error) is within the well of convergence (pink). [**NR**] $\boxed{\text{chapter1/. cycle-skipping}}$



(a)



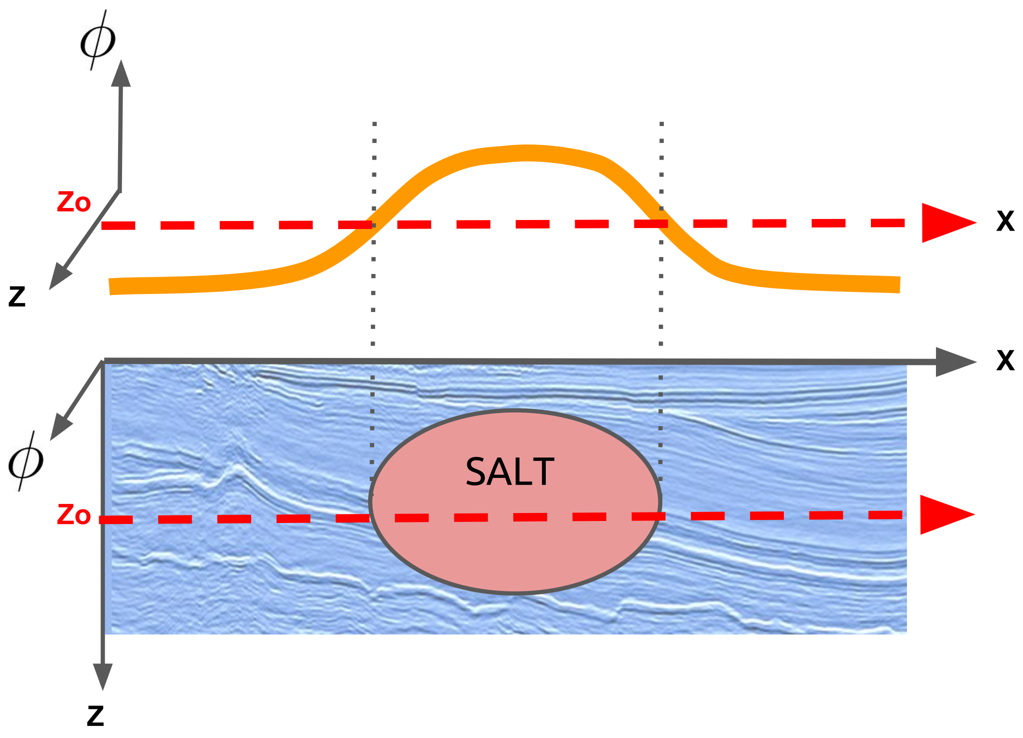
(b)

Figure 1.5: a) Legacy velocity model. b) Velocity model after FWI updating Shen et al. (2017). [**NR**] chapter1/. xukai-true,xukai-fwi

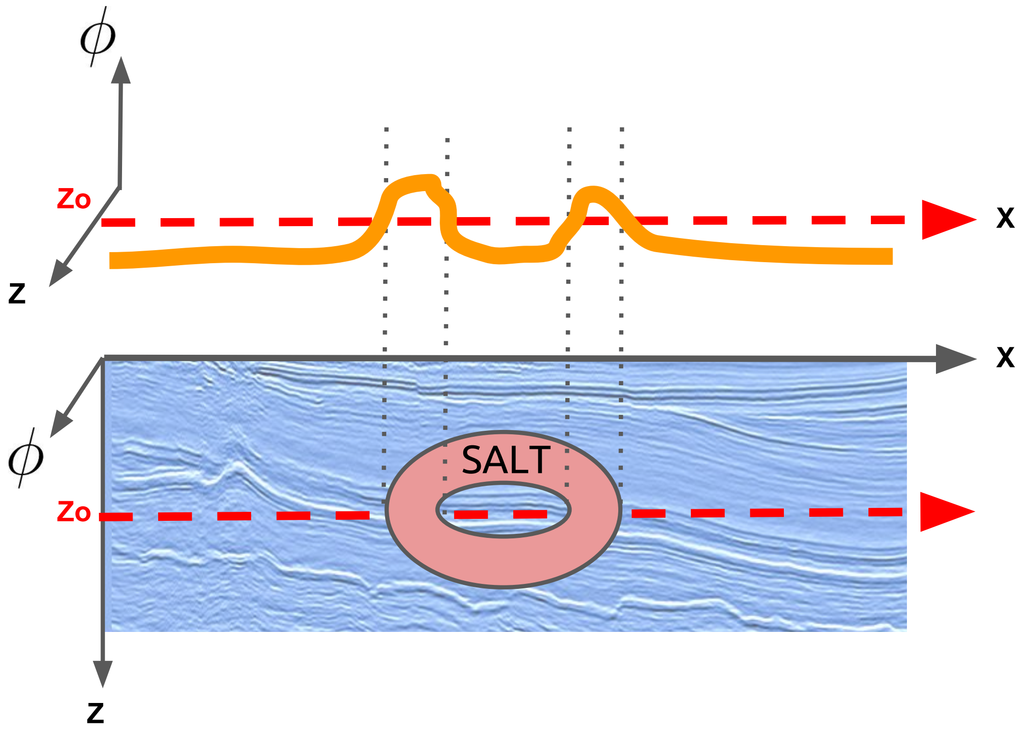
modeling problem, the salt boundary can be represented using a level set as in Figures 1.6(a) and 1.6(b). With this representation, one can easily modify the shape and topology of the level set *implicitly* by modifying its higher dimensional surface (for this reason, called an implicit surface). While explicit boundary tracking schemes break down for sharp edges or topology changes, level sets very elegantly manage these cases. For any problem where we wish to track the boundary of a homogeneous (or approximately homogeneous) body, level sets are an ideal tool. They have found use in fields like aerospace engineering for modeling fluid flow around airfoils (Xia et al., 2010), as well as image segmentation problems in medicine (Maciejewski et al. (2012),Tsai et al. (2003)). For the purposes of salt modeling, they give us the advantage of adjusting and updating a sharp boundary position (high spatial frequency content), even when the updates from our data contain only low frequency content (see Figure 1.7). Since cycle skipping is based on the spectra of the data and not the velocity model, we can gain the advantage of manipulating high spatial frequency boundaries while still mitigating the risk of cycle skipping by skewing the spectra of our data space to low frequencies. Furthermore, computing wave propagations for low frequencies is cheaper than propagating high frequencies.

What has already been done?

Level sets are an excellent tool for tracking boundaries, and their debut was ushered in by the work of Osher and Sethian (1988). They are especially useful for the class of problems called *shape optimization* that optimize an objective function based on shape properties. Santosa (1996) provides an early demonstration of solving these



(a)



(b)

Figure 1.6: Diagram showing a cross section (orange curve) through a 3D implicit surface that represents the salt boundary in the 2D model (bottom panel) for both a) circular salt and b) donut-hole models . [**NR**] chapter1/. levelset-cross-section,levelset-cross-section-donut

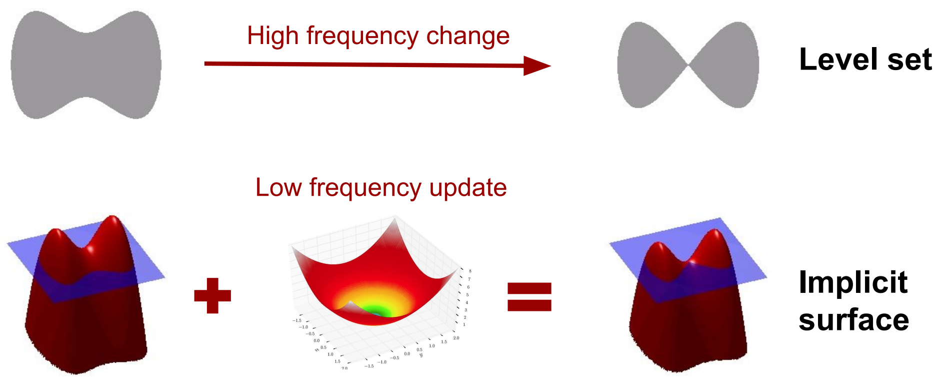


Figure 1.7: Diagram demonstrating how low spatial frequency updating leads to a change in a high spatial frequency boundary represented by a level set contour. [**NR**] chapter1/. levelset-update

types of problems with an L2 norm objective function, namely for deconvolution and diffraction-screen reconstruction problems. Burger (2003) continued to build a more unified theory of level set methods for inverse problems. Later work by Lewis et al. (2012) and Guo and de Hoop (2013) applied shape optimization to the FWI objective function for the purpose of segmenting salt bodies. Guo and de Hoop (2013) utilize this approach with frequency domain wave propagation to evolve a salt boundary and velocity model. However, these approaches use the full model domain to represent the level set, and do not explore the advantages of using second-order updating. More recently, Kadu et al. (2016) introduced the use of radial basis functions to sparsify the model space and speed computation for second order updating, but did not extend the work to 3D models or field data examples.

What does this thesis add?

In this work, I claim to demonstrate success using shape optimization with level sets to build 3D models of salt bodies at depth using field data. The differentiating aspects of this work include a number of advances. First, the use of radial basis functions (RBFs) to reduce the number of model parameters has been improved by distributing them with spatial variance according to a given likelihood of salt updating. I leverage the sparsified model that this representation yields to to speed our inversion of the Newton step that our Gauss-Newton Hessian approximates. I further leverage interpreter guidance as input by molding the initial shape of our implicit surface according to the same spatial likelihood function that our RBF centers are based on. Last, I further extend the practice by applying this method to a 3D ocean bottom node (OBN) dataset.

Chapter 2

Level sets and shape optimization

In the previous chapter I reviewed the problems with modeling salt and how level sets are an ideal tool for addressing some of those challenges. In this chapter I explain the basics of level sets and how they apply to the shape optimization problem of finding a salt boundary that minimizes the FWI objective function. I derive the gradient and demonstrate updating on a simple 2D model. Last, I derive the Hessian equations and demonstrate updating on 2D models with a comparison between the full Hessian and the Gauss-Newton Hessian approaches.

LEVEL SETS AS A TOOL

Basic definitions

A level set is a contour of a higher dimensional surface, ϕ. One can use the level set as an elegant tool for keeping track of boundaries as they change form. If one considers a 2D level set being the contour of a 3D implicit surface, then the spatial domain of the 2D level set can be defined as $\Theta \subset \mathbf{R^2}$ with elements $\chi \in \Theta$. A salt body Ω is simply a subset of the 2D domain ($\Omega \subset \Theta$) such that:

$$\Omega = \{\chi \mid \phi(\chi, \tau) > 0\},$$

where τ indicates the 'time' axis along which the evolution steps progress ($\tau = 0$ is the initial iteration). As such, for a single updating step (along τ), our current salt body Ω evolves to Ω'. I further define the boundary of the salt body as Γ (see Figure

2.1), such that:

$$\Gamma = \{\chi \mid \phi(\chi, \tau) = 0\}.$$

I define a point along the boundary curve to be:

$$\chi_\Gamma \in \Gamma.$$

With this definition of the boundary points, the level set of ϕ that represents the salt body boundary can be described as:

$$\phi(\chi_\Gamma, \tau) = 0.$$

While this derivation uses a 2D salt model for simplicity, a 3D salt body would instead have a four-dimensional implicit surface.

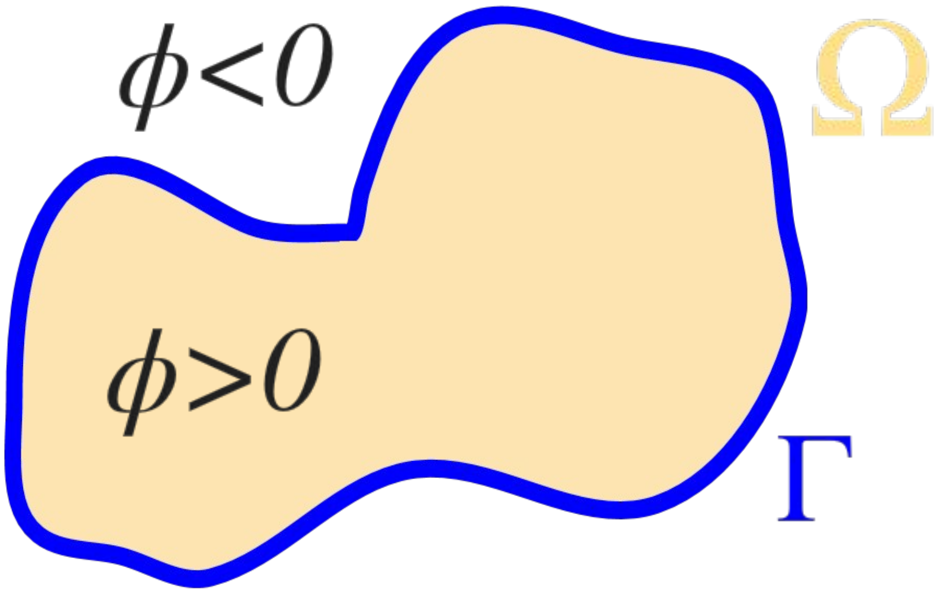


Figure 2.1: Diagram explaining the salt body domain (Ω), the ϕ values inside and outside of the salt, and the salt boundary (Γ). [**NR**] chapter2/. levelset-domain

Shape optimization derivation

With a clear understanding of level sets being established, the first step of the shape optimization derivation is to define the objective function we wish to minimize. I choose a variation of the FWI objective function

$$\psi(\mathbf{p}) = \frac{1}{2}||\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}||_2^2, \tag{2.1}$$

Here $\mathbf{F}$ is the forward acoustic wave propagator which includes a source function, wave propagation, and sampling based on a receiver geometry. Our seismic velocity model is $\mathbf{m}$, and $\mathbf{d_{obs}}$ is the observed pressure data. I deviate from the traditional FWI objective function by parameterizing $\mathbf{m}$ in terms of $\mathbf{p}$. Here I define $\mathbf{p} = [\boldsymbol{\phi} \quad \mathbf{b}]^T$, with $\boldsymbol{\phi}$ as the implicit surface function and $\mathbf{b}$ as the background velocity function, both varying across the full spatial domain Θ. In this section, I intend to take the derivative of equation 2.1 with respect to the underlying parameters $\mathbf{b}$ and $\boldsymbol{\phi}$. Since the chain rule applies, I first take the derivative with respect to $\mathbf{m}$ and then $\mathbf{p}$. I start by expanding our definition of the objective function:

$$\psi(\mathbf{p}) = \frac{1}{2}||\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}||_2^2 \tag{2.2}$$

$$= \frac{1}{2}(\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}})^T(\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}) \tag{2.3}$$

$$= \frac{1}{2}(\mathbf{F}(\mathbf{m}(\mathbf{p}))^T\mathbf{F}(\mathbf{m}(\mathbf{p})) - 2\mathbf{F}(\mathbf{m}(\mathbf{p}))^T\mathbf{d_{obs}} + \mathbf{d_{obs}}^T\mathbf{d_{obs}}), \tag{2.4}$$

I then take the first derivative:

$$\frac{d\psi}{d\mathbf{p}} = \frac{d\mathbf{m}(\mathbf{p})}{d\mathbf{p}}^T \frac{d\mathbf{F}(\mathbf{m}(\mathbf{p}))}{d\mathbf{m}}^T \mathbf{F}(\mathbf{m}(\mathbf{p})) - \frac{d\mathbf{m}(\mathbf{p})}{d\mathbf{p}}^T \frac{d\mathbf{F}(\mathbf{m}(\mathbf{p}))}{d\mathbf{m}}^T \mathbf{d_{obs}} \tag{2.5}$$

$$= \frac{d\mathbf{m}(\mathbf{p})}{d\mathbf{p}}^T \frac{d\mathbf{F}(\mathbf{m}(\mathbf{p}))}{d\mathbf{m}}^T (\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}) \tag{2.6}$$

$$= \frac{d\mathbf{m}(\mathbf{p})}{d\mathbf{p}}^T \mathbf{B}^T(\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}). \tag{2.7}$$

Here one recognizes $\frac{d\mathbf{F}(\mathbf{m}(\mathbf{p}))}{d\mathbf{m}}^T$ as the familiar adjoint Born operator ($\mathbf{B}^T$) used in standard FWI (Virieux and Operto (2009)). However, we must continue to expand our gradient to be defined in terms of our underlying parameters $\mathbf{b}$ and $\boldsymbol{\phi}$ (contained in vector $\mathbf{p}$). This requires us to state our model space clearly:

$$\mathbf{m}(\mathbf{p}) = \mathbf{m}(\boldsymbol{\phi}, \mathbf{b}) = H(\boldsymbol{\phi})(\mathbf{c}_{\text{salt}} - \mathbf{b}) + \mathbf{b}, \tag{2.8}$$

where $H(\circ)$ is the Heaviside function. Because I assume a homogeneous salt body, $\mathbf{c}_{\text{salt}}$ is a constant salt-velocity vector, typically about 4500m/s. Figure 2.2 explains this equation as the sum of a salt overlay created from $\boldsymbol{\phi}$ with the background velocity model $\mathbf{b}$. Note that the background velocity (shown in Figure 2.2c) is defined even in the region under the salt footprint. That way if the salt shrinks to a smaller size or otherwise retracts inward, the background velocity is already defined and does not need to be interpolated.

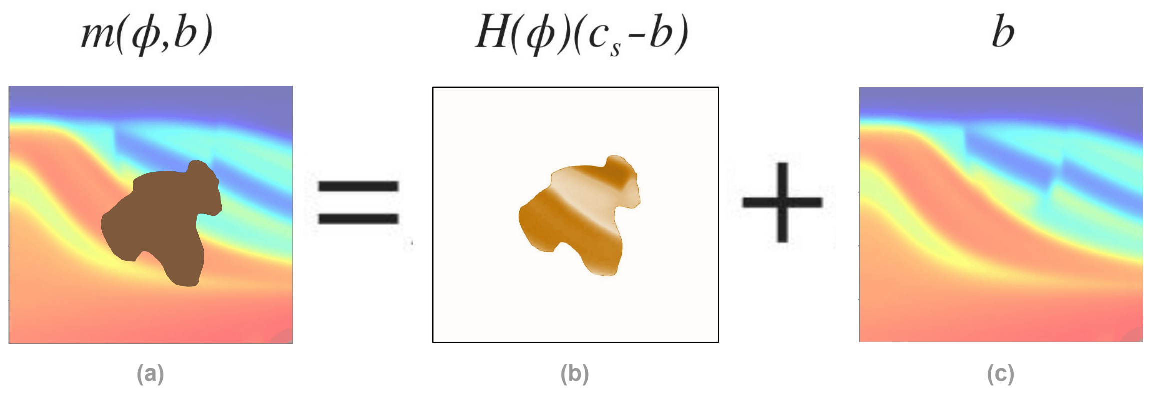


Figure 2.2: a) Full velocity model. b) Salt overlay. c) Background velocity. [**NR**] chapter2/. model-components

I expand the definition in equation 2.8 with a Taylor series as:

$$\mathbf{m_1} = \mathbf{m_0} + \frac{\partial \mathbf{m}}{\partial \boldsymbol{\phi}}\Big|_{\mathbf{m_0}} \triangle\boldsymbol{\phi} + \frac{\partial \mathbf{m}}{\partial \mathbf{b}}\Big|_{m_0} \triangle\boldsymbol{b} + ... \tag{2.9}$$

This approximation is only valid when the Taylor series converges with the addition of increasingly higher order terms. For the Heaviside function, this is not the case, since the function is not differentiable in its original form. For this reason, I use a smoothed approximation of the Heaviside function, such as:

$$\widetilde{H}(\phi) = \frac{1}{2}\left[1 + \frac{2}{\pi}\arctan(\frac{\pi\phi}{\epsilon})\right]. \tag{2.10}$$

An advantage of using equation 2.10 as a Heaviside approximation is that the derivative is non-zero everywhere. One downside is that the support of the function is infinite, and it will only approach +1 or 0 at + inf or − inf. Further, the derivative of this function is very high near the zero crossing point. Kadu et al. (2017b) introduces an alternate approximation that has compact support in the region surrounding the zero-crossing:

$$\widetilde{H}_\epsilon(\phi) = \begin{cases} 0 & \text{if } \phi \ < \ -\epsilon, \\ \frac{1}{2}\left[1 + \frac{\phi}{\epsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\phi}{\epsilon}\right)\right] & \text{if } -\epsilon \ \leq \ \phi \ \leq \ \epsilon, \\ 1 & \text{if } \phi \ > \ \epsilon. \end{cases} \tag{2.11}$$

Besides being able to capture +1 and 0 values exactly, the derivative of this approximation varies much less (see curves in Figure 2.3) and better balances weighting to the full boundary region when applying level set updates. By substituting this formulation of the Heaviside function in equation 2.8, I can now truncate the series in equation 2.9 and ignore higher order terms. This creates a linear approximation for the perturbation of the velocity model $\mathbf{m}$ with respect to $\boldsymbol{\phi}$ and $\mathbf{b}$:

$$\triangle \boldsymbol{m} \approx \frac{\partial \mathbf{m}(\boldsymbol{\phi}_o, \mathbf{b}_o)}{\partial \boldsymbol{\phi}} \triangle \boldsymbol{\phi} + \frac{\partial \mathbf{m}(\boldsymbol{\phi_o}, \boldsymbol{b_o})}{\partial \mathbf{b}} \triangle \boldsymbol{b}. \tag{2.12}$$

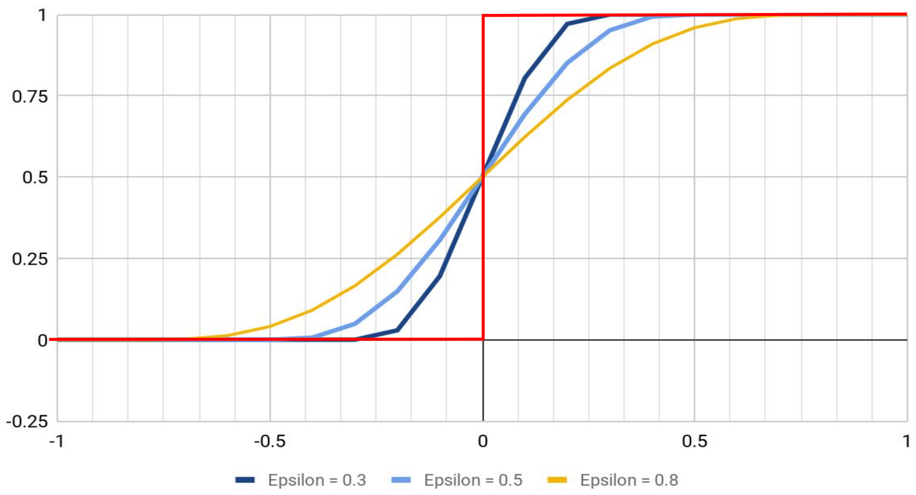


Figure 2.3: Curves of Heaviside approximations based on equation 2.11 for varying values of ϵ, and true Heaviside function (red). Note that within $\{-\epsilon, \epsilon\}$, the slope of each curve (its derivative) is relatively constant. [**ER**] $\boxed{\text{chapter2/. heaviside-approx-chart}}$

This can be written as a matrix operation:

$$\begin{aligned}
\triangle \boldsymbol{m} &\approx \begin{bmatrix} \frac{\partial \mathbf{m}(\boldsymbol{\phi_o},\mathbf{b_o})}{\partial \boldsymbol{\phi}} & \frac{\partial \mathbf{m}(\boldsymbol{\phi_o},\mathbf{b_o})}{\partial \mathbf{b}} \end{bmatrix} \begin{bmatrix} \triangle \boldsymbol{\phi} \\ \triangle \boldsymbol{b} \end{bmatrix} \\
\triangle \boldsymbol{m} &\approx \begin{bmatrix} \frac{\partial \mathbf{m}(\boldsymbol{\phi_o},\mathbf{b_o})}{\partial \boldsymbol{\phi}} & \frac{\partial \mathbf{m}(\boldsymbol{\phi_o},\mathbf{b_o})}{\partial \mathbf{b}} \end{bmatrix} \triangle \boldsymbol{p},
\end{aligned}$$

where I define operator $\mathbf{D}$ as:

$$\begin{aligned}
\mathbf{D} &= \begin{bmatrix} \frac{\partial \mathbf{m}(\boldsymbol{\phi_o},\mathbf{b_o})}{\partial \boldsymbol{\phi}} & \frac{\partial \mathbf{m}(\boldsymbol{\phi_o},\mathbf{b_o})}{\partial \mathbf{b}} \end{bmatrix} \\
&= \begin{bmatrix} \widetilde{\delta}(\boldsymbol{\phi_o})(\mathbf{c}_{\text{salt}} - \mathbf{b_o}) & \mathbf{I} - \widetilde{H_\epsilon}(\boldsymbol{\phi_o}) \end{bmatrix}. \qquad (2.13)
\end{aligned}$$

Here, $\widetilde{H_\epsilon}$ is the Heaviside approximation (equation 2.11), $\mathbf{I}$ is the identity matrix, $\widetilde{\delta}$ is the derivative of $\widetilde{H_\epsilon}$, $\mathbf{b}$ is the background velocity ($\mathbf{b_0}$ is fixed), $\boldsymbol{\phi}$ is the implicit surface ($\boldsymbol{\phi_0}$ is fixed), and $\mathbf{c}_{salt}$ is the constant salt velocity. Its functional form, $\widetilde{\delta}(\cdot)$ approximates an impulse function at $\phi = 0$, and its application acts as a selector for the salt boundary. Because of this, the operator $\mathbf{D}$ ultimately scales and masks the parameter fields $\triangle\boldsymbol{\phi}$ and $\triangle\boldsymbol{b}$.

This new approximation of the perturbation can be combined in our velocity model with equation 2.7 to get:

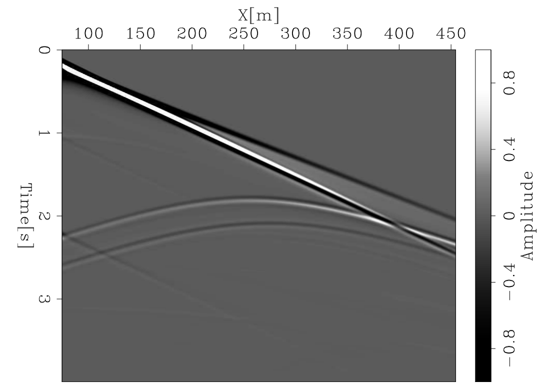
$$\begin{aligned}
\frac{d\psi}{d\mathbf{p}} &= \frac{d\mathbf{m}(\mathbf{p})}{d\mathbf{p}}^T \mathbf{B}^T(\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}) \qquad (2.14) \\
&\approx \mathbf{D}^T\mathbf{B}^T(\mathbf{F}(\mathbf{m}(\mathbf{p})) - \mathbf{d_{obs}}) \qquad (2.15) \\
&\approx \mathbf{D}^T\mathbf{B}^T\triangle\boldsymbol{d}. \qquad (2.16)
\end{aligned}$$

SIMPLE 2D SYNTHETIC MODEL DEMONSTRATION

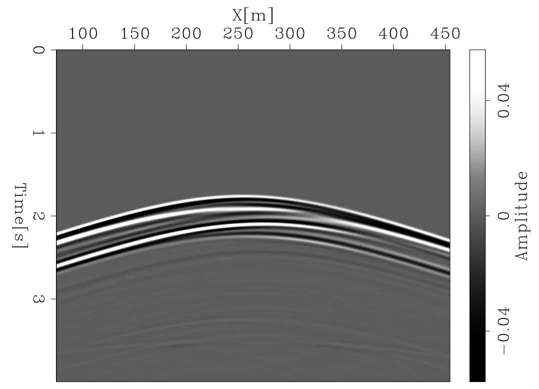
In order to demonstrate first order (gradient) updating on ϕ (assuming $\mathbf{b}$ is constant for simplicity), I apply the inversion workflow shown in Algorithm 1 on a 2D circular salt model example (Figure 2.8(a)), with an outward normal perturbation (the initial salt is larger than the true model). This creates an error in the modeled data (Figure 2.4(a)) and thus a data space residual (Figure 2.4(b)). Note that these figures show the top and bottom salt events between about 1.7-2.5 seconds.

Algorithm 1 Steepest descent updating algorithm

```
1: procedure LevelSetInversion-order1( $d_{\text{obs}}$,$\phi_0$ )
2:     for $i$ in (1,$N$) do
3:         $d_{\text{syn}}(i) = \mathrm{F}(\phi_{i-1})$
4:         residual($i$) = $d_{\text{obs}} - d_{\text{syn}}(i)$
5:         gradient($i$) = $D^T B^T$residual($i$)
6:         $\alpha$ = linesearch(gradient($i$))
7:         $\phi_i = \phi_{i-1} - \alpha \cdot$ gradient($i$)
8:     end for
9:     Return $m(\phi_N)$
10: end procedure
```



(a)



(b)

Figure 2.4: a) Observed data. b) Data residuals. Observed data and residuals are a single shot gather created from the small initial salt example (Figure 2.12(a)). [**ER**] chapter2/. obsdata-sample-small,residuals-sample-small

I calculate the search direction for $\boldsymbol{\phi}$ (Figure 2.6) by back-propagating the data residual and cross-correlating it with the source wavefield as per the Born approximation imaging condition (Figure 2.5), followed by applying the **D** operator from equation 2.13. In this case, the search direction is negative near the perturbation, since it wants to decrease what is ultimately a positive velocity error. This search direction pushes a decrease in the value of the implicit surface (compare Figures 2.7(a) and 2.7(b)). This decrease draws the zero-level set deeper so that it is in closer alignment with the true salt boundary. Applying the approximate Heaviside function (equation 2.11) to this updated implicit surface (Figure 2.7(b)) gives us a new model that is closer in form to the true model (Figure 2.8(b)). This process can be continued, making iterative updates to the implicit surface and as a result, the velocity model itself.

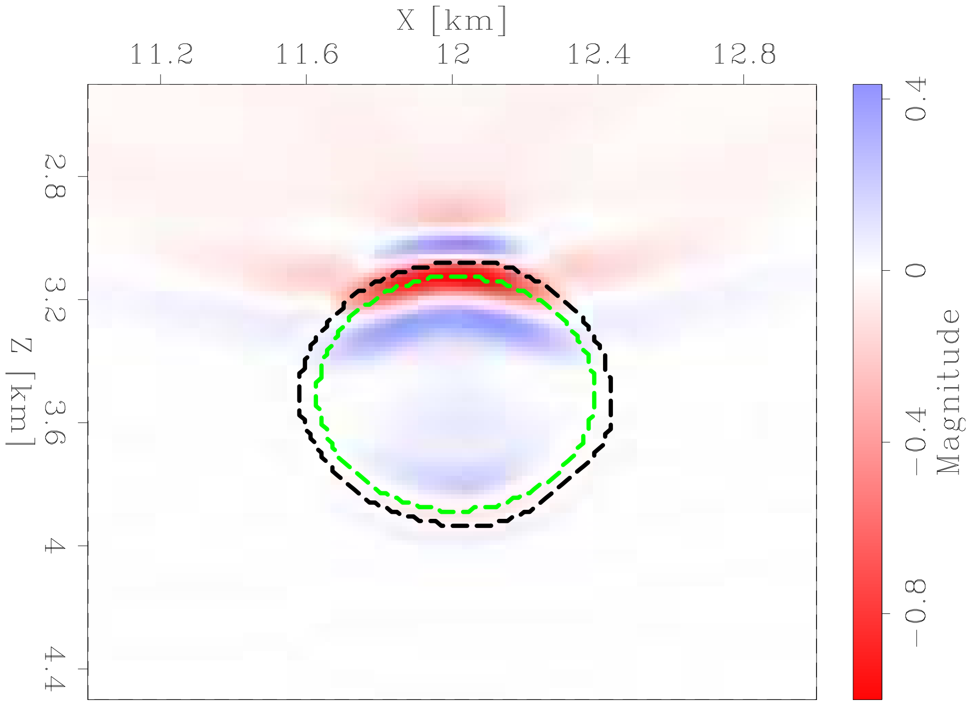


Figure 2.5: Search direction from adjoint Born image of back-propagated residuals used in conventional FWI. True salt extent (green dashed line), initial salt extent (black dashed line). [**CR**] chapter2/. rtmGrad0-big

For an initial model where the salt circle is too small (Figure 2.12(a)), the adjoint Born component of the search direction (Figure 2.9) is positive, since it is trying to increase the model velocity to that of the salt. The full search direction for $\boldsymbol{\phi}$ gives a positive perturbation of the implicit surface (Figure 2.10). This update raises the implicit surface (compare Figures 2.11(a) and 2.11(b)), moving the salt boundary upwards to correct for the deep boundary discrepancy (Figure 2.12(b)).

INTRODUCING THE HESSIAN OPERATOR

We can use the Newton method (equation 2.17) to perform the inversion, which helps remove the effect of our operator from the gradient, improves the search direction, and subsequently speeds up convergence. This is done by inverting the Hessian ($\mathbf{H}$) of the objective function we are minimizing and applying it to the negative of our gradient ($\mathbf{g}$) to find the search direction $\triangle \mathbf{m}$:

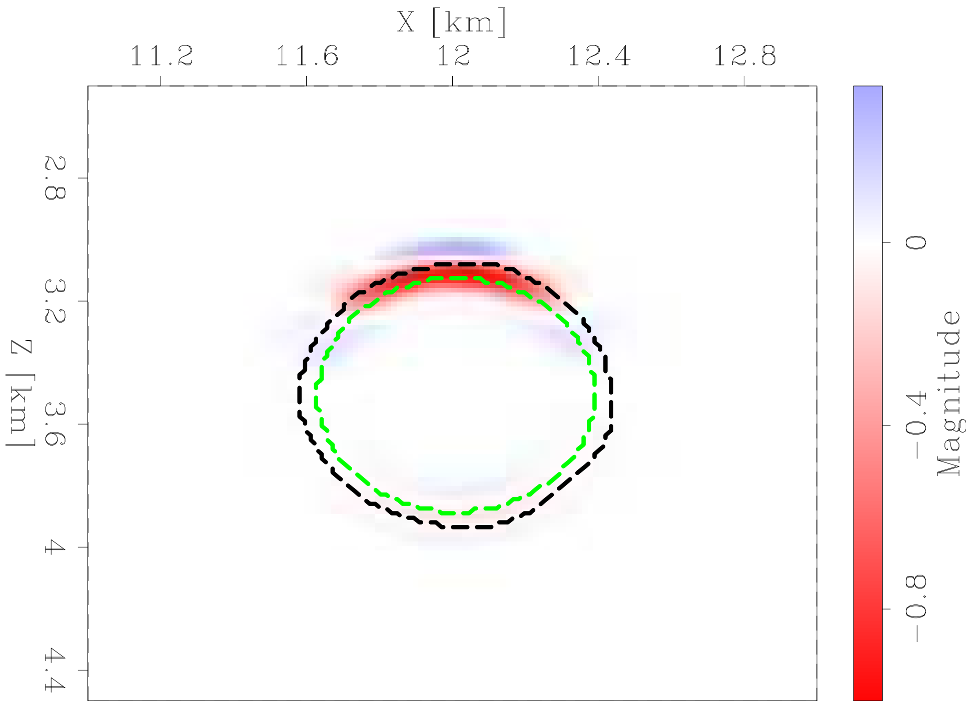


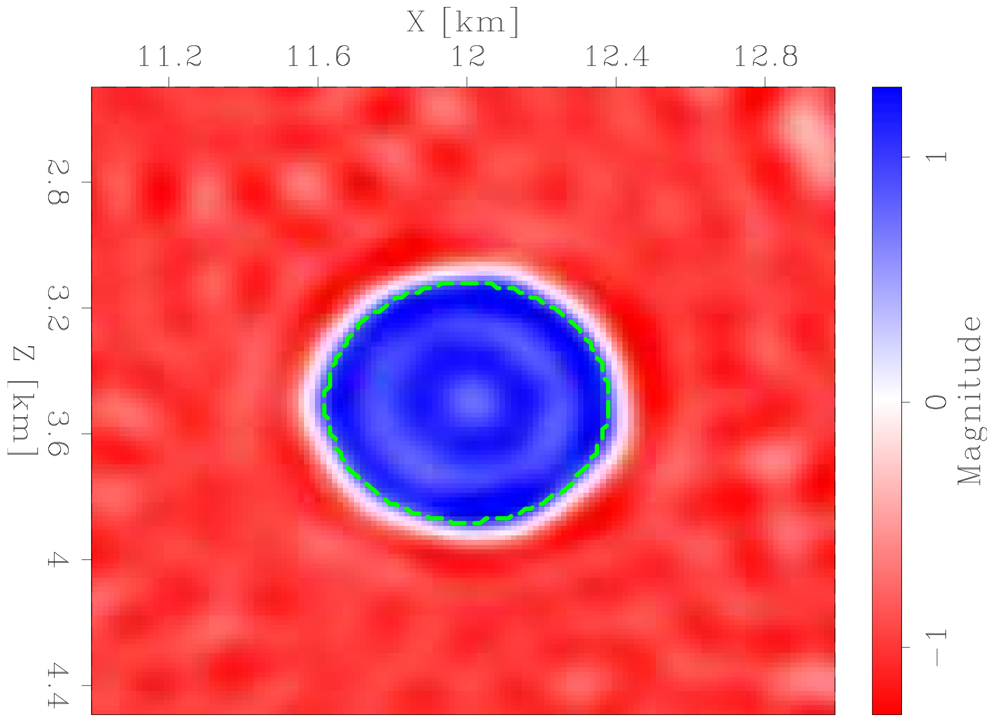
Figure 2.6: Implicit surface search direction ($\triangle\boldsymbol{\phi}$). True salt extent (green dashed line), initial salt extent (black dashed line). [**CR**] $\boxed{\text{chapter2/. phiGrad0-big}}$

$$\triangle\mathbf{m} = -\mathbf{H}^{-1}\mathbf{g}. \tag{2.17}$$

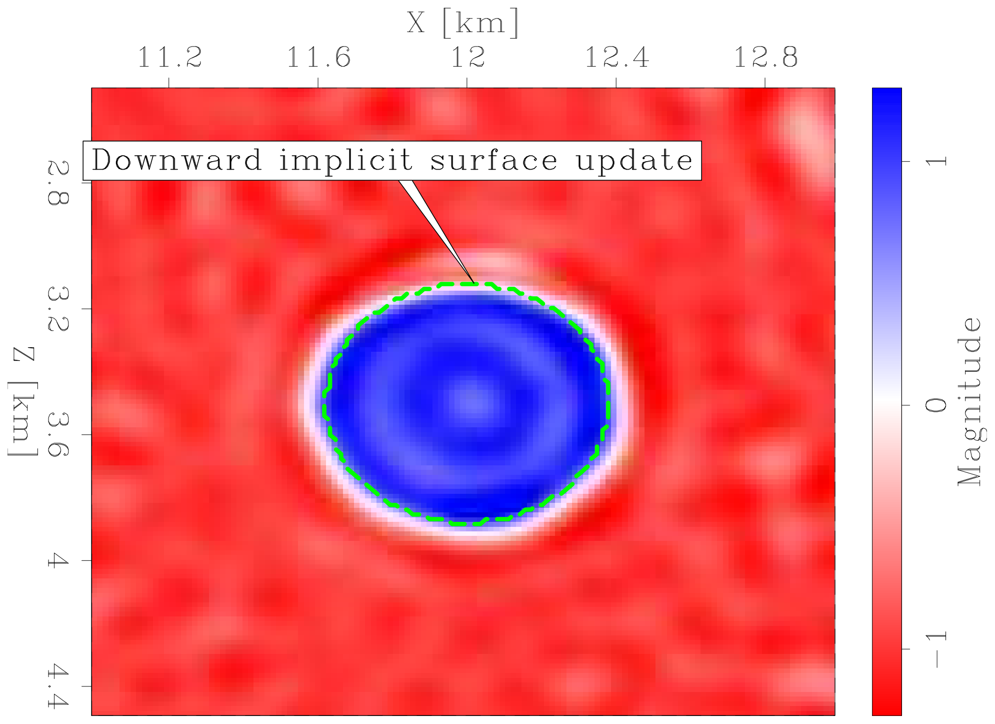
For our case, we can represent the Hessian as the second derivative (equation 2.18) of the FWI objective function (equation 2.1):

$$\begin{aligned}
\frac{\delta\psi}{\delta\mathbf{m}} &= \frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}^T}(\mathbf{F}(\mathbf{m}) - \mathbf{d_{obs}}) \\
\frac{\delta}{\delta\mathbf{m}}\frac{\delta\psi}{\delta\mathbf{m}} &= \frac{\delta}{\delta\mathbf{m}}\frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}^T(\mathbf{F}(\mathbf{m}) - \mathbf{d_{obs}}) \\
\frac{\delta^2\psi}{\delta\mathbf{m}^2} &= \frac{d^2\mathbf{F}(\mathbf{m})}{d\mathbf{m}^2}^T(\mathbf{F}(\mathbf{m}) - \mathbf{d_{obs}}) + \frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}^T\frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}.
\end{aligned} \tag{2.18}$$

However, this second derivative contains a term based on the data residuals. This term can be expensive to compute and is often neglected, so the Gauss-Newton approximation is used instead:

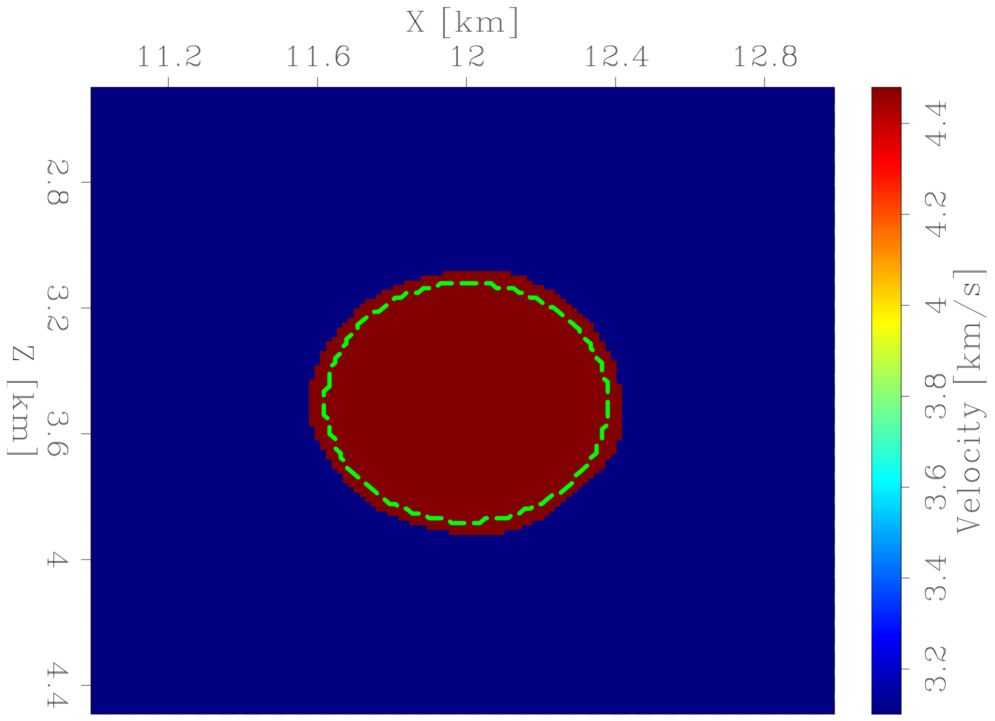


(a)

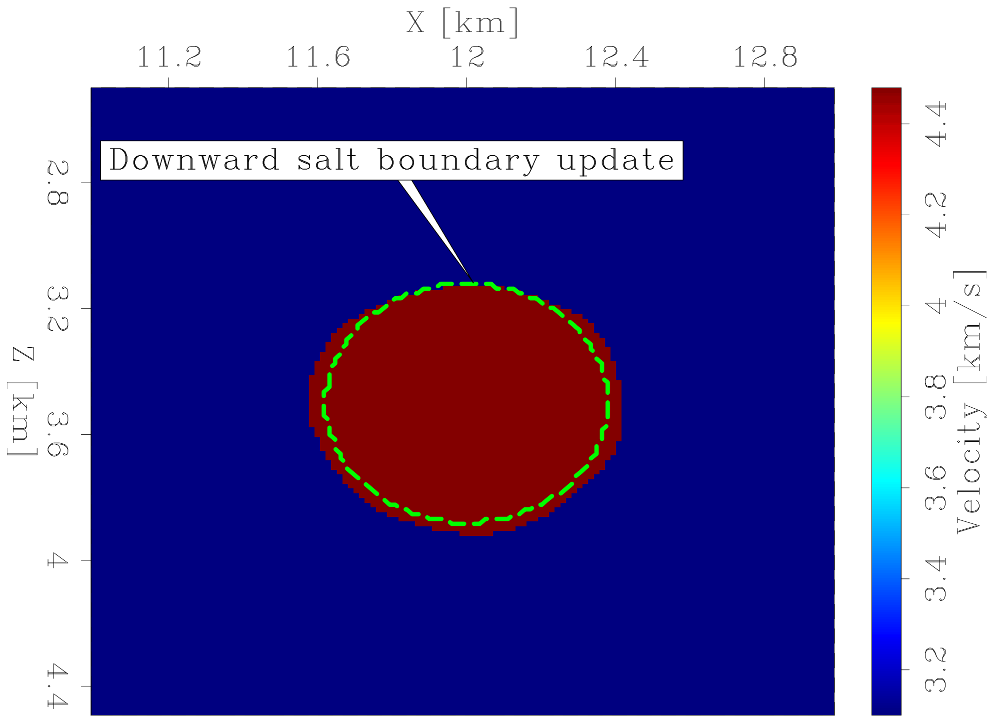


(b)

Figure 2.7: Implicit surface (ϕ) model a) before, and b) after updating with search direction (Figure 2.6). True salt extent shown as green dashed line. [**CR**] chapter2/. initphi-big,nextphi-big



(a)



(b)

Figure 2.8: Velocity model (**m**) a) before, and b) after updating. True salt extent shown as green dashed line. [**CR**] chapter2/. initmodel-big,nextmodel-big

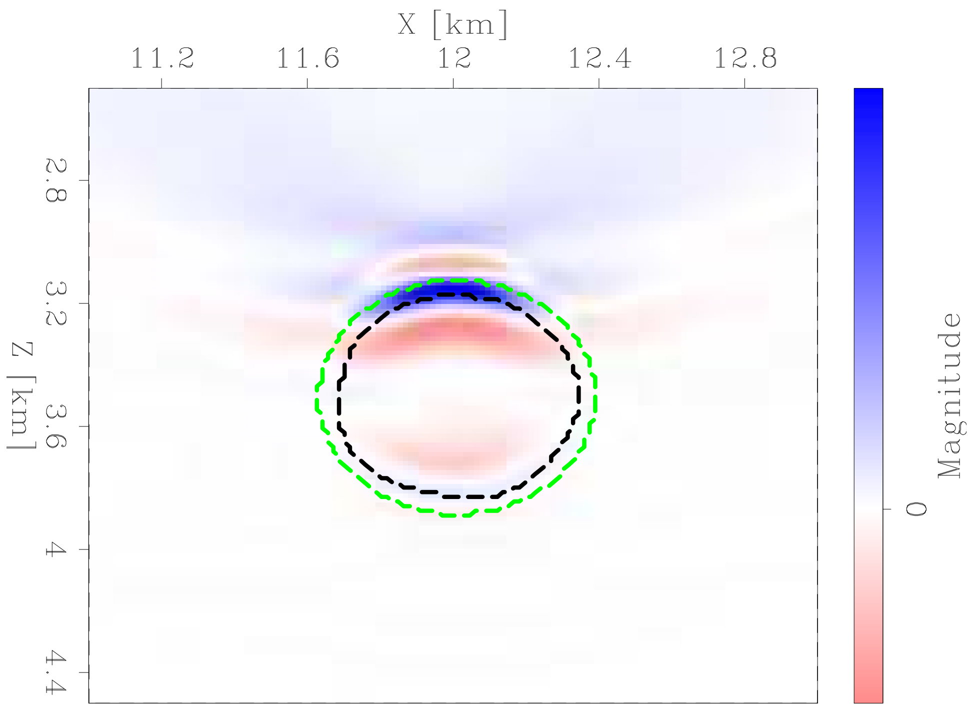


Figure 2.9: Search direction from adjoint Born image of back-propagated residuals used in conventional FWI. True salt extent (green dashed line), initial salt extent (black dashed line). [**CR**] chapter2/. rtmGrad0-small

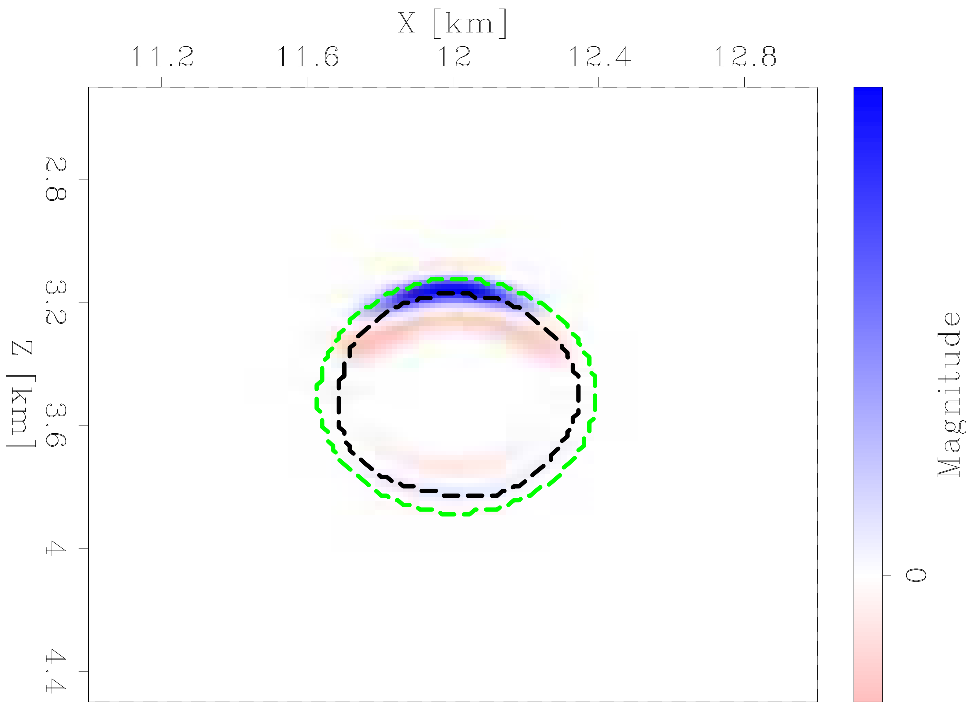
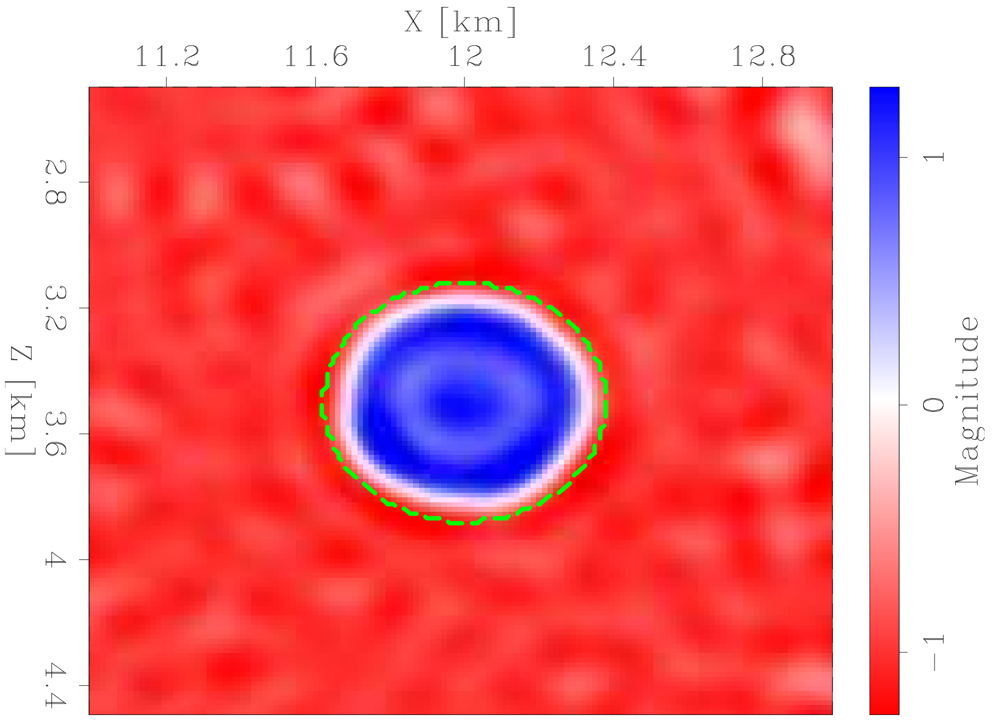
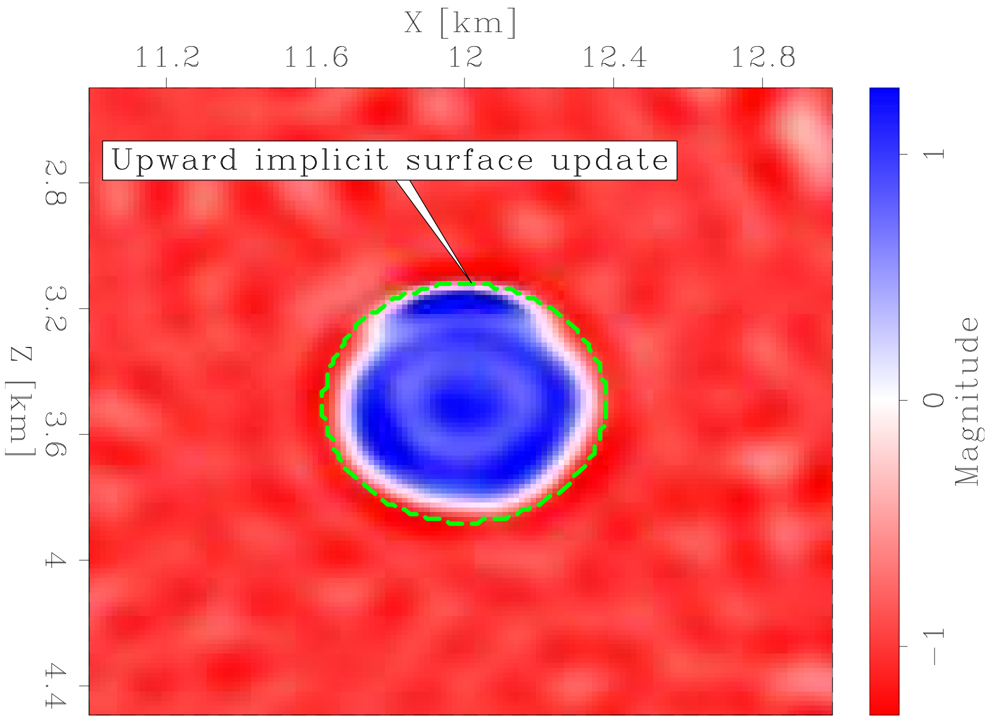


Figure 2.10: Implicit surface search direction ($\triangle\phi$). True salt extent (green dashed line), initial salt extent (black dashed line). [**CR**] chapter2/. phiGrad0-small

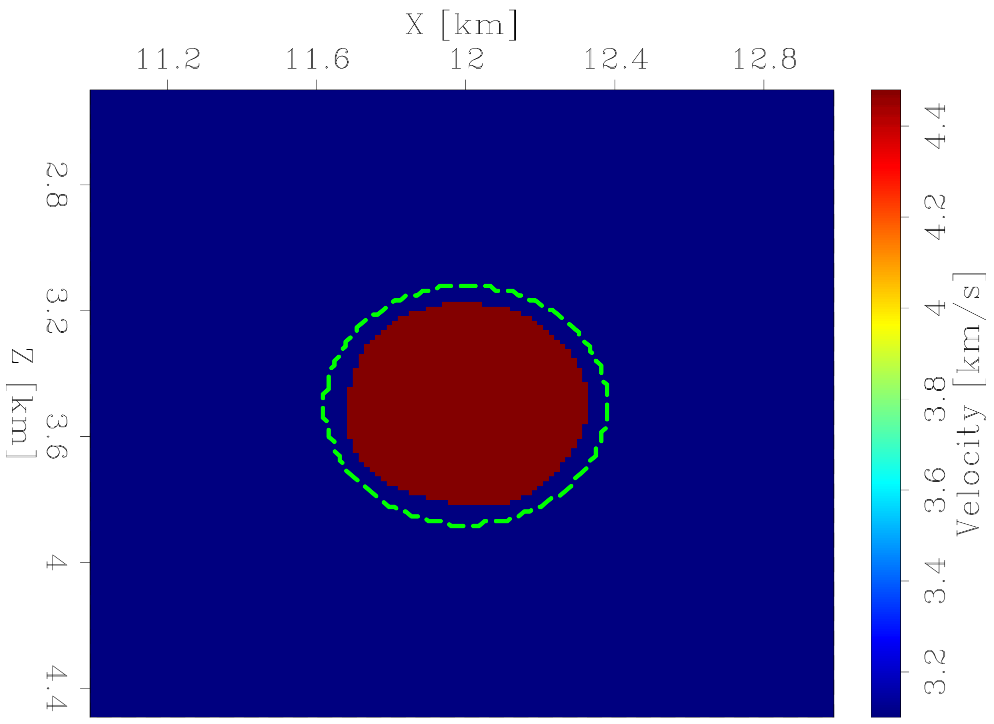


(a)

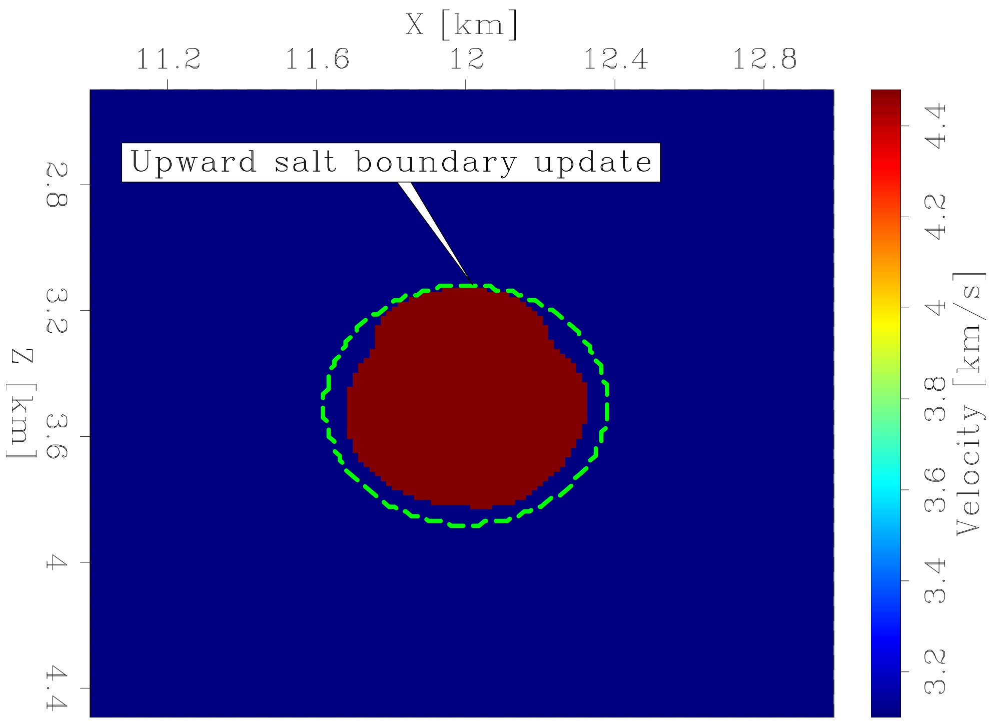


(b)

Figure 2.11: Implicit surface (ϕ) model a) before, and b) after updating with search direction (Figure 2.10). True salt extent shown as green dashed line. [**CR**] chapter2/. initphi-small,nextphi-small



(a)



(b)

Figure 2.12: Velocity model (**m**) a) before, and b) after updating. True salt extent shown as green dashed line. [**CR**] chapter2/. initmodel-small,nextmodel-small

$$\begin{aligned}\frac{\delta^2\psi}{\delta\mathbf{m}^2} &= \frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}^T\frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}} + \frac{d^2\mathbf{F}(\mathbf{m})}{d\mathbf{m}^2}^T(\mathbf{F}(\mathbf{m}) - \mathbf{d}_{\mathbf{obs}}) \\ \mathbf{H}_{\mathbf{GN}} &= \frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}^T\frac{d\mathbf{F}(\mathbf{m})}{d\mathbf{m}}\end{aligned} \tag{2.19}$$

While using this approximation is cheaper to compute, it is not as accurate. Fichtner (2010) and Biondi et al. (2015) show that the residual term of the FWI objective function Hessian can be found by computing WEMVA-like operations on the data-space residuals. By comparison, the Gauss-Newton approximation of the FWI objective function is simply the forward Born operator followed by the adjoint Born operator. An inherent limitation of the Born operator is that it only models first-order reflections, and so double-scattered energy such as waves that bounce inside canyons or salt bodies (Figure 2.13) can be spatially misplaced in the search direction when using the Gauss-Newton Hessian. One would expect to gain a better search direction for canyon and salt-type models by using the full Hessian instead. For this reason, I investigate using the full Hessian on 2D synthetic models.

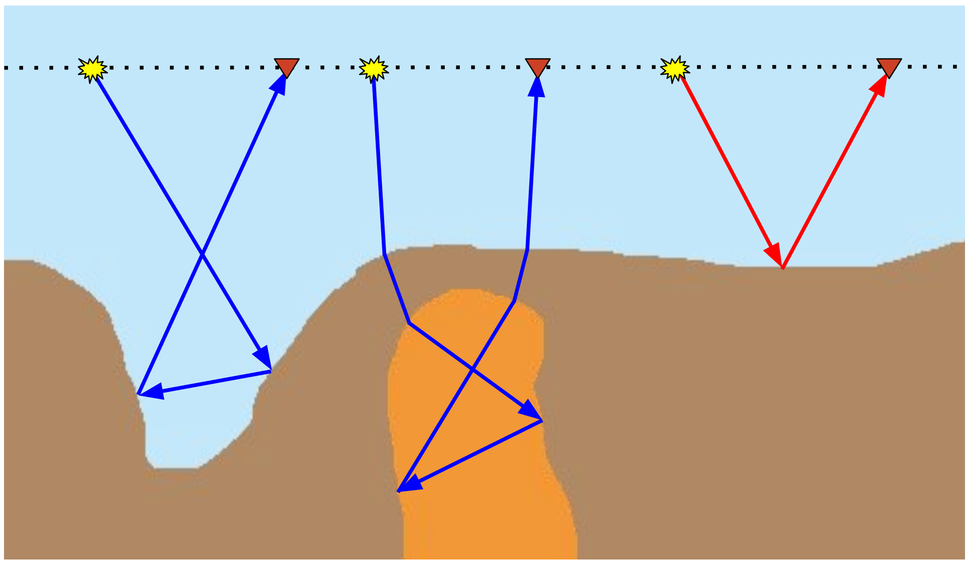


Figure 2.13: Second order reflection ray paths in a canyon and inside a salt body (blue). First order reflection off water bottom (red). **[NR]** chapter2/. reflection-order-diagram

HESSIAN INVERSION COMPARISONS

As discussed in the last section, the full Hessian is potentially better than the Gauss-Newton Hessian at finding a search direction for models that contain salt canyons. I test this hypothesis by creating a synthetic salt canyon true model, and then perturb it to create a starting model so that a true search direction is known. Then I run separate inversions with the full and Gauss Newton Hessians to compare search direction results. Furthermore, I first test by perturbing only one side of the canyon, and then test by perturbing both sides. This allows us to see how the starting velocity model affects the efficacy of the Gauss-Newton and full Hessians relative to one another.

Single canyon perturbation example

For the first example, I select an upper canyon portion of the synthetic Sigsbee model (Figure 2.14). I perturb the left hand side of the canyon (Figure 2.15) to create secondary scattering against the opposite canyon wall (shown in Figure 2.16). I use an evenly spaced acquisition geometry of 38 shots and 230 receivers, and perform modeling using absorbing boundary conditions and a Ricker wavelet with a central frequency of 15 Hz. For both the single and double perturbation cases, I assume that $\triangle \mathbf{b} = 0$, and so invert for a model defined as $\triangle \mathbf{p} = \triangle \boldsymbol{\phi}$ (inverting only for the implicit surface update, not the background velocity update).

Double canyon perturbation example

For the second example, I use the same true model as before (Figure 2.14). However, this time I perturb both the left and right hand sides of the canyon (Figure 2.17). This offers further complexity to the secondary scattering of the model (shown in Figure 2.18). The same acquisition geometry and wavelet were used.

Benefits of the full Hessian

When comparing the results of the Gauss-Newton (Figure 2.19) and the full Hessian results (Figure 2.20) from the double canyon perturbation model, one can see a slight improvement in the focusing of the energy in the full Hessian example. This improved

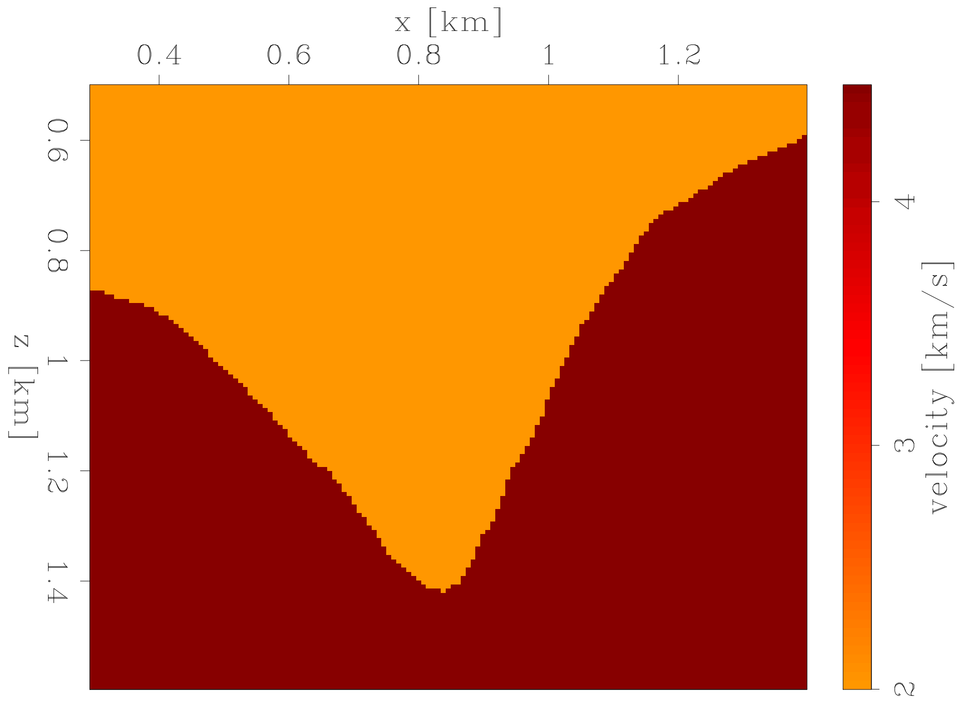


Figure 2.14: The canyon portion of the Sigsbee model that was used. [**ER**] chapter2/. single-guess

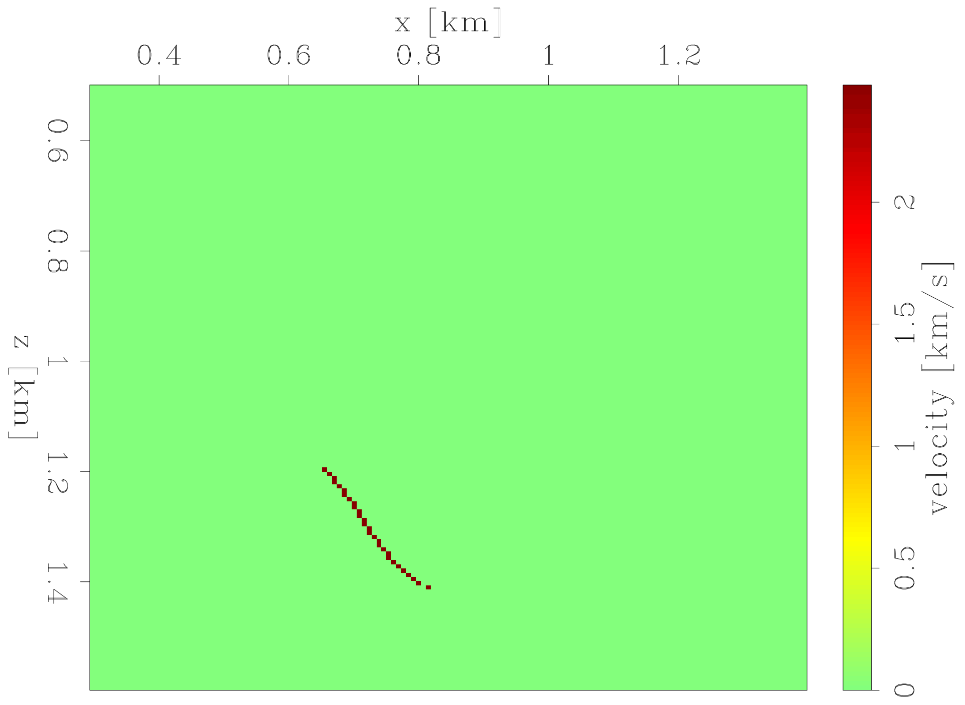


Figure 2.15: The single canyon perturbation ($\triangle \mathbf{m}_{\text{actual}}$) of the Sigsbee model. [**ER**] chapter2/. single-pert

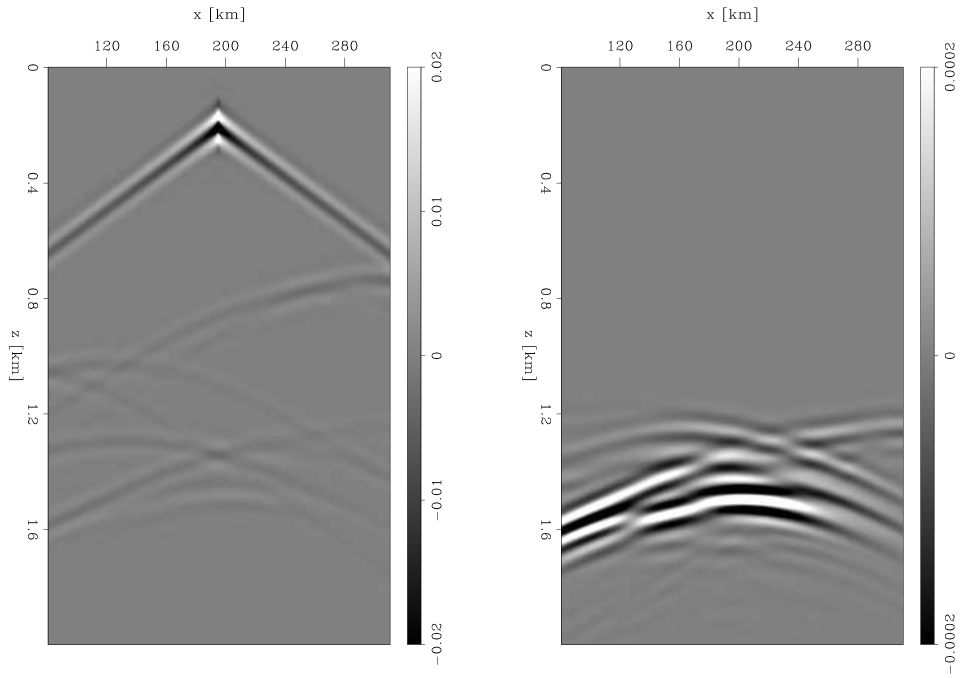


Figure 2.16: Data generated on the true model ($\mathbf{d_{obs}}$) from the center shot (left). The data residual ($\triangle\mathbf{d}$) for the same center shot (right) generated by differencing $\mathbf{d_{obs}}$ with data generated from the single perturbation model. [**CR**] chapter2/. centershot-analysis-single

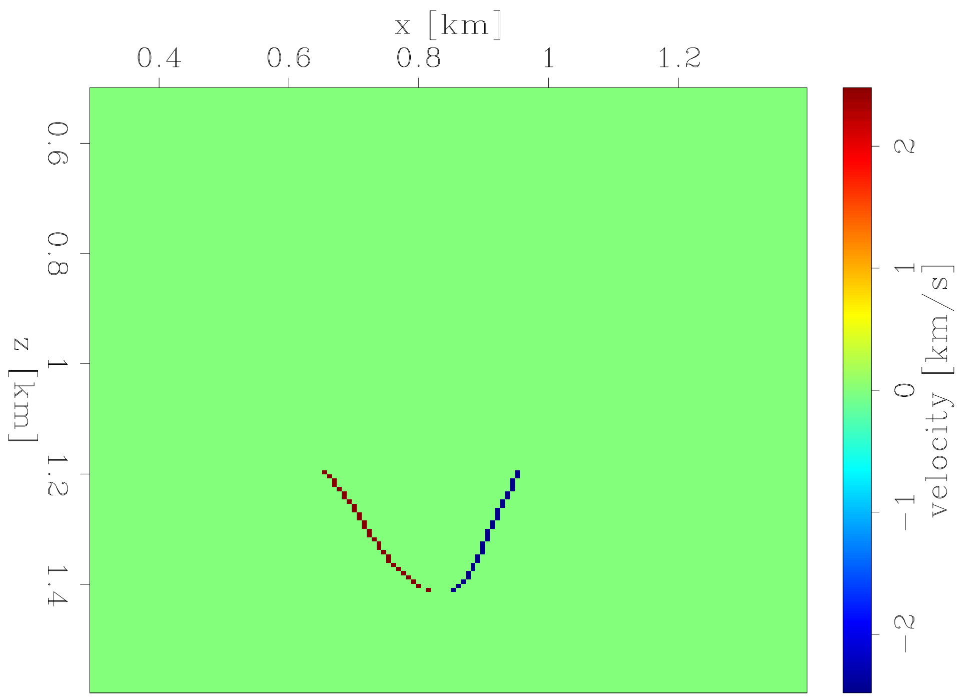


Figure 2.17: The double canyon perturbation ($\triangle \mathbf{m}_{\text{actual}}$) of the Sigsbee model. [**ER**] $\boxed{\text{chapter2/. double-pert}}$

search direction should lead to better convergence in the greater non-linear inversion scheme. When comparing against the steepest descent search direction for the double perturbation case (Figure 2.21), one can see that the search directions by either type of Hessian inversion create significant improvements. The improvement of the full Hessian versus the Gauss-Newton Hessian that is seen in the double perturbation case is minimal compared to the improvement that the Gauss-Newton Hessian has over the steepest descent direction (Figure 2.21). Similarly in the single perturbation case, the search direction from inverting the Gauss-Newton Hessian (Figure 2.22) is significantly better than the steepest descent search direction (Figure 2.23).

Limitations

However, the single perturbation case demonstrates that the advantages of the full Hessian are not realized for all models, since the Hessian is model dependent. The single perturbation example results are much different with regards to the full Hessian

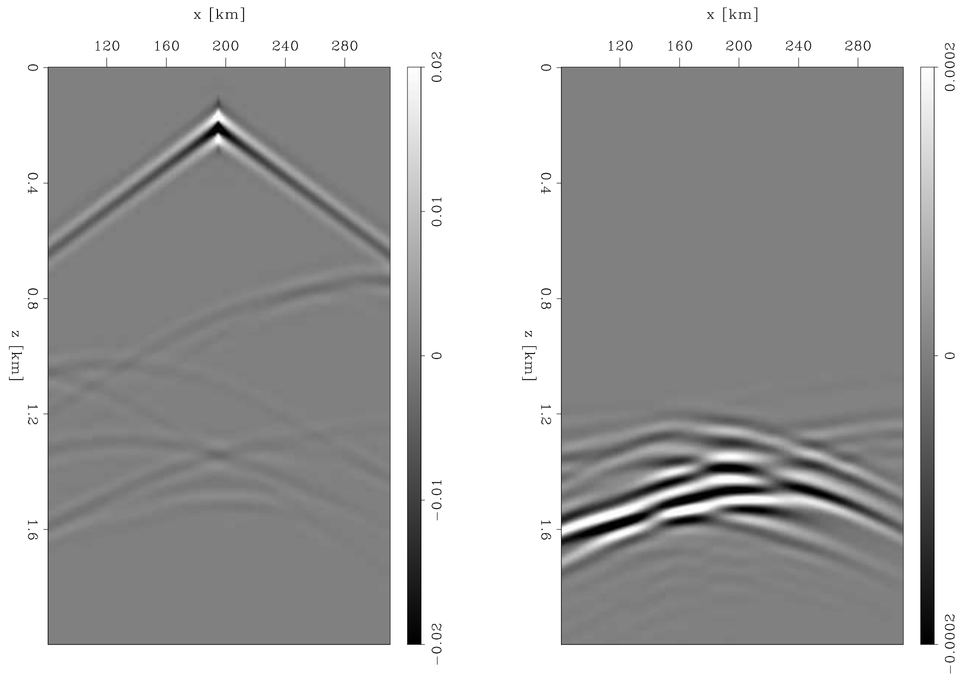


Figure 2.18: Data generated on the true model ($\mathbf{d_{obs}}$) from the center (left). The data residual ($\triangle \mathbf{d}$) for the same center shot (right) generated by differencing $\mathbf{d_{obs}}$ with data generated from the double perturbation model. [**CR**] chapter2/. centershot-analysis-double

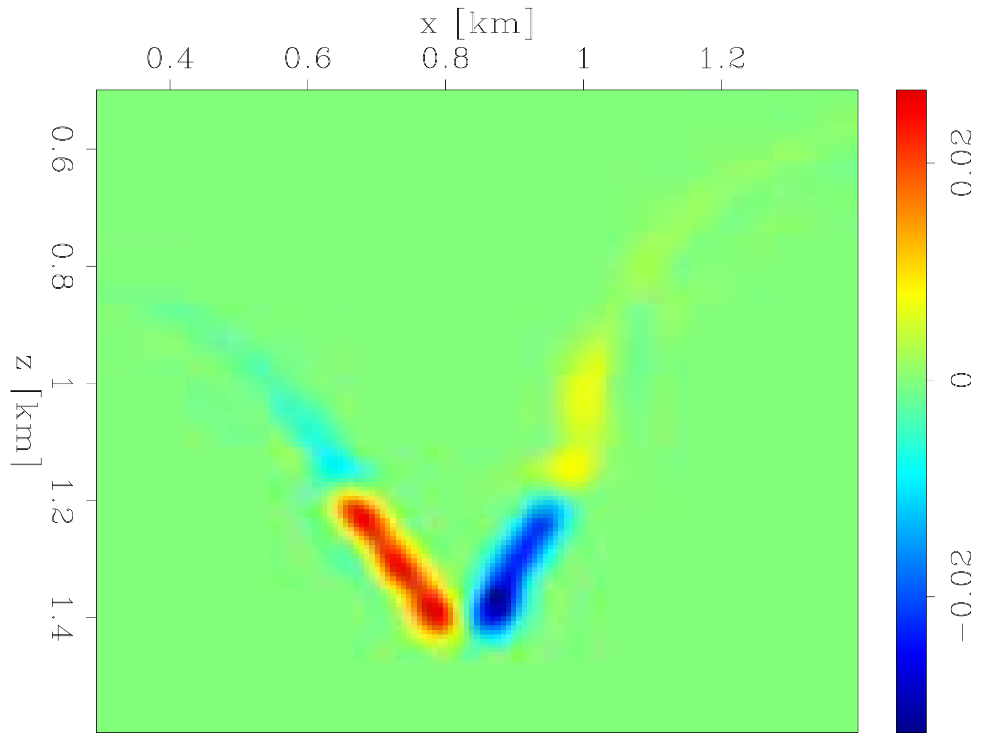


Figure 2.19: The Newton search direction ($\triangle\phi$) using the inverted Gauss-Newton approximation of the Hessian on the double perturbation model. [**CR**] chapter2/. double-final-gn

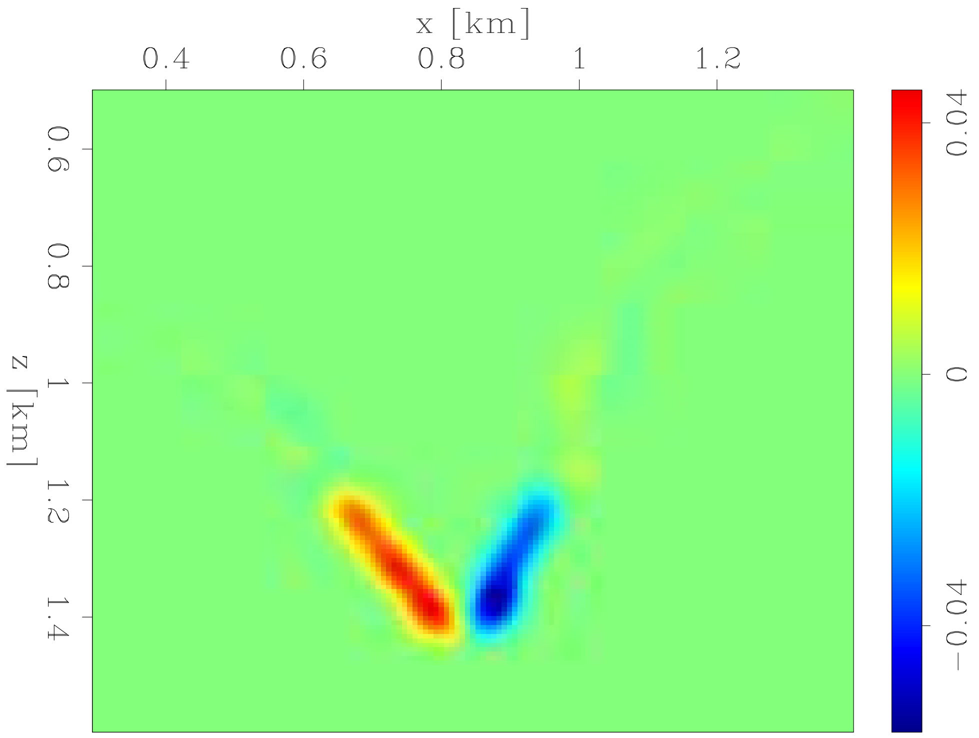


Figure 2.20: The Newton search direction ($\triangle\boldsymbol{\phi}$) using the inverted full Hessian on the double perturbation model. [**CR**] $\boxed{\text{chapter2/. double-final-full}}$

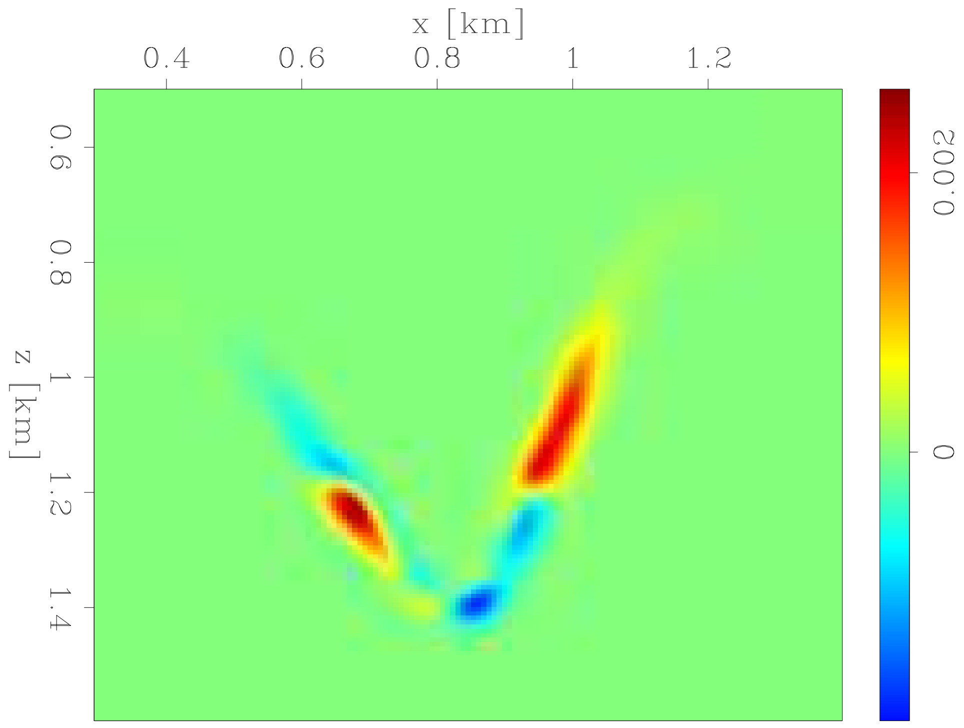


Figure 2.21: The steepest descent search direction ($\triangle\boldsymbol{\phi}$) (negative of the FWI gradient) for the double perturbation model. [**CR**] $\boxed{\text{chapter2/. double-negative-gradient}}$

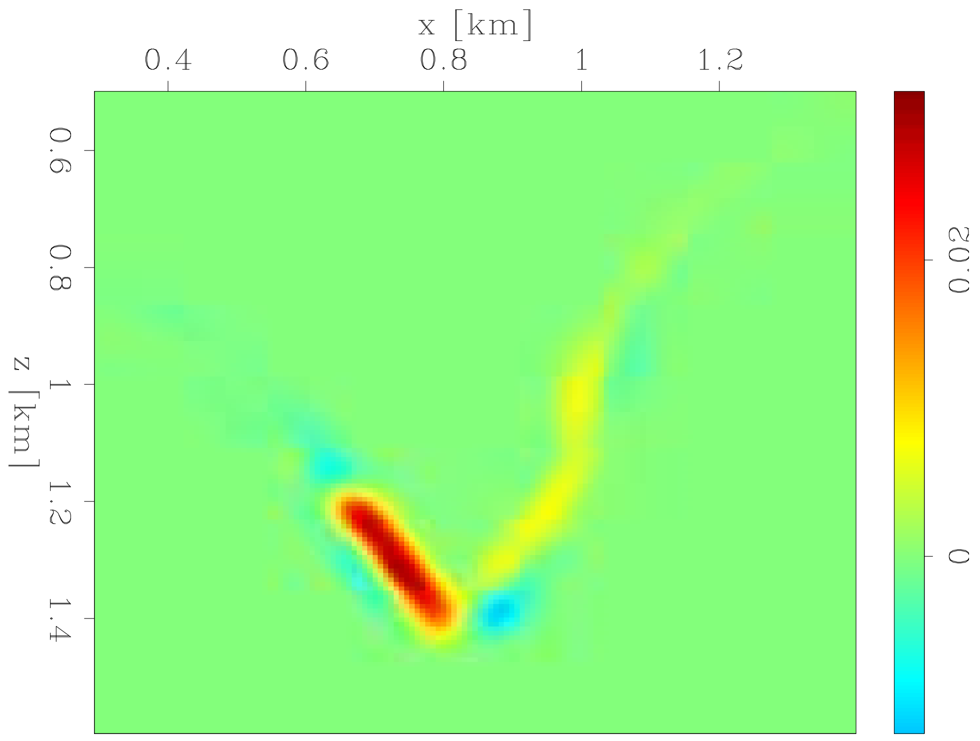


Figure 2.22: The Newton search direction ($\triangle\phi$) using the inverted Gauss-Newton approximation of the Hessian on the single perturbation model. [**CR**] chapter2/. single-final-gn

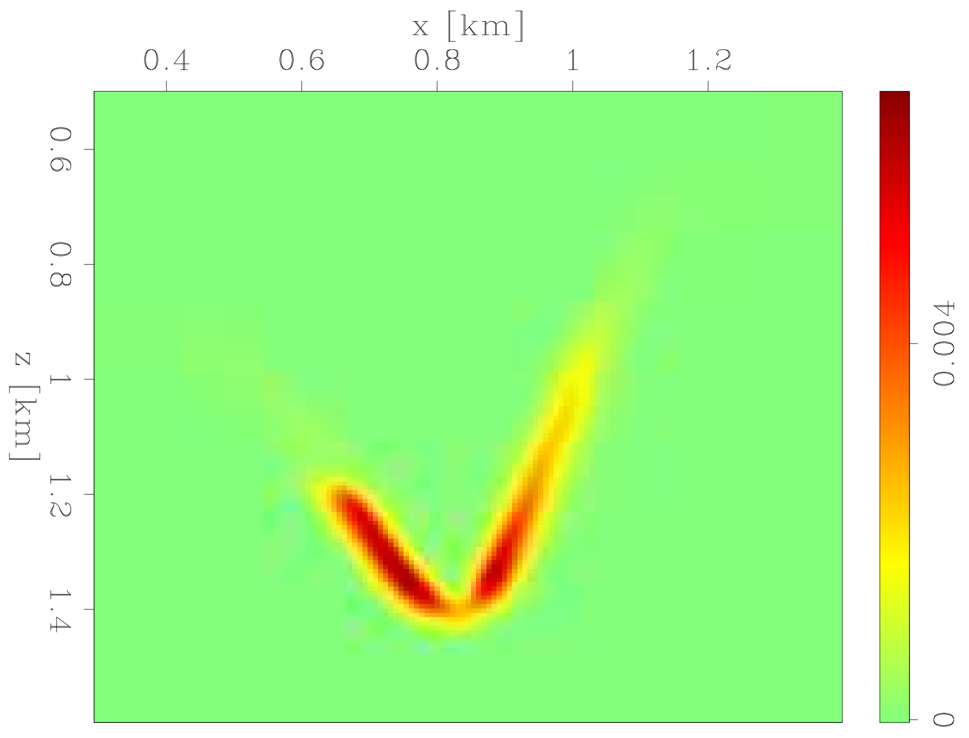


Figure 2.23: The steepest descent search direction ($\triangle\boldsymbol{\phi}$) (negative of the FWI gradient) for the single perturbation model. [**CR**] $\boxed{\text{chapter2/. single-negative-gradient}}$

search direction. While the Gauss-Newton Hessian system inversion rapidly converges (Figures 2.22 and 2.25(a)), the full Hessian inversion becomes unstable part way through (Figures 2.24 and 2.25(b)). Because the full Hessian operator is not inherently positive semi-definite like the Gauss-Newton Hessian, it may have negative eigenvalues, which can lead to instability during inversion. This was very likely the case in the single canyon perturbation example. On the other hand, with the double perturbation example we get stable convergence using either method (compare Figures 2.26(a) and 2.26(b)).

One of the few feasible methods for enforcing our operator to be positive semi-definite is to use the Levenberg-Marquardt (1963) method of regularizing the operator with a scaled identity matrix:

$$\hat{\mathbf{H}}_{\mathbf{full}} = \mathbf{H}_{\mathbf{full}} + \alpha\mathbf{I} \tag{2.20}$$

However, in order to use this method properly, the correct scaling of the identity

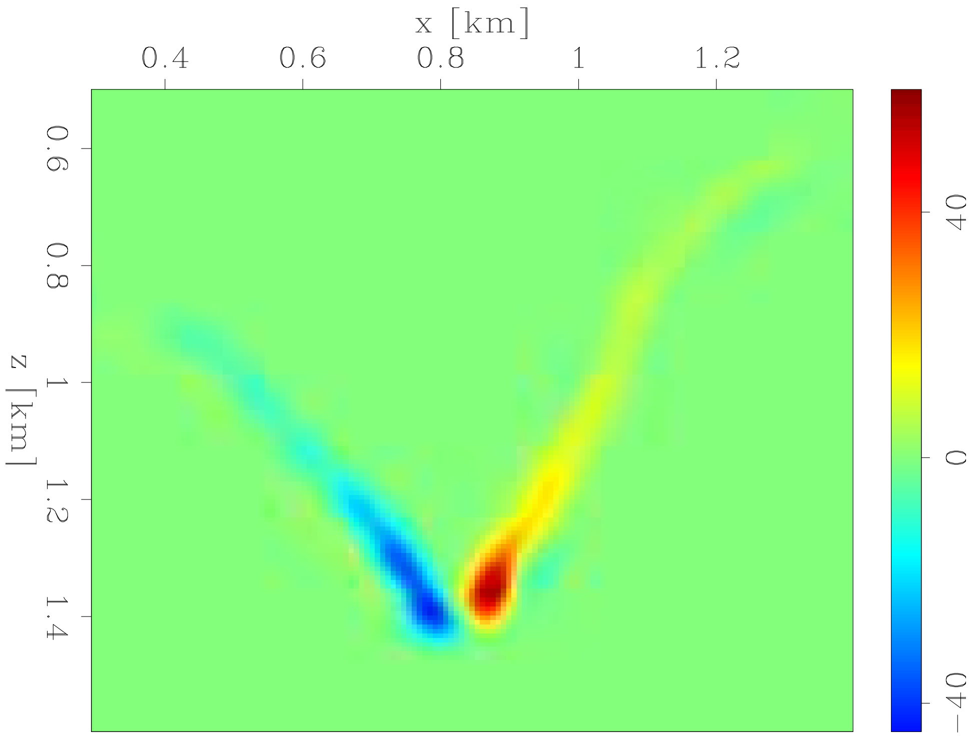
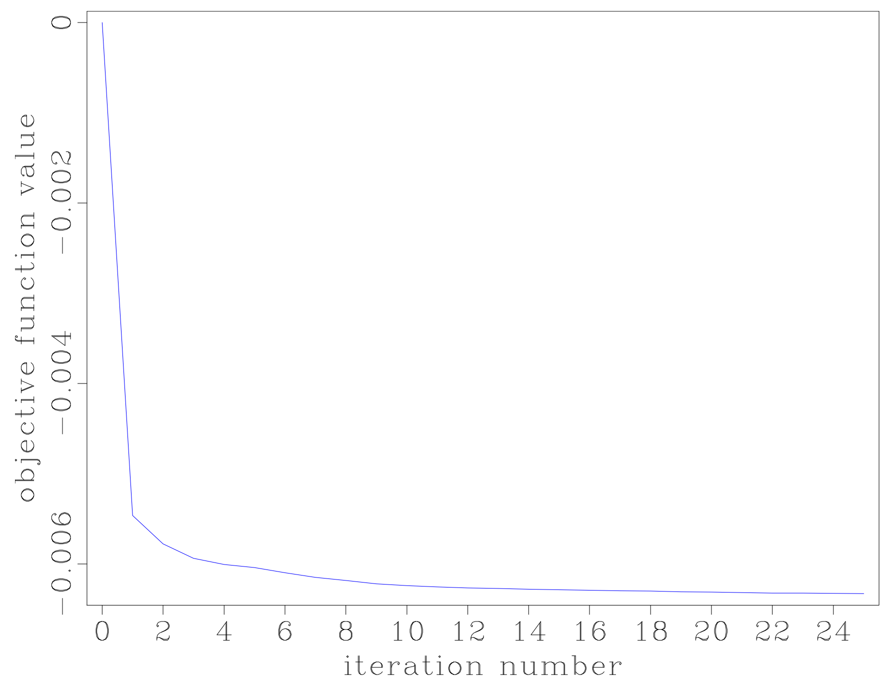


Figure 2.24: The Newton search direction using the inverted full Hessian on the single perturbation model. [**CR**] chapter2/. single-final-full

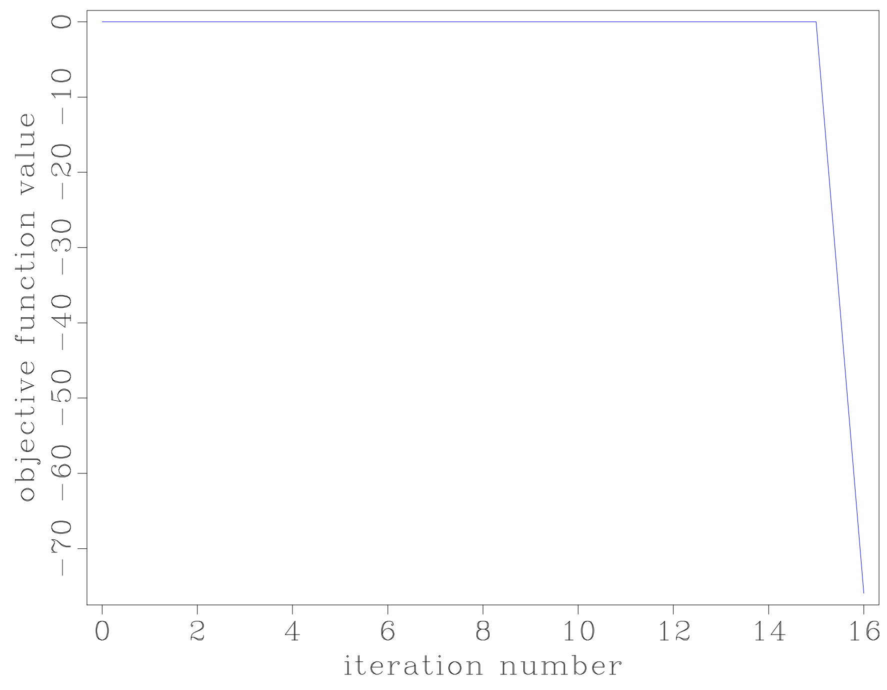
matrix must be found. If too large of a scaling is selected, the operator becomes more like a scaled identity matrix, negating the potential benefit of inverting the full Hessian system to begin with. If the scaling is too small, the system will still be ill-conditioned, and prone to instability as observed earlier. The ideal scaling is slightly more than the value of the most negative eigenvalue of the operator. This makes the operator positive definite. Since our model (and as a result, our Hessian) is very large, it is impractical to store or factorize the Hessian matrix to determine the most negative eigenvalue through traditional non-iterative linear algebra methods.

Power Iteration Method

The most practical way to find the best scaling is by using the power iteration method outlined in Larson (2009) to find the maximum absolute-valued eigenvalue (positive in the case shown for Figure 2.27). After this has been found, one shifts the diagonal of the operator by the negative of this value, and then repeat the power iterations to find a new maximum absolute-valued eigenvalue. The difference between this value and

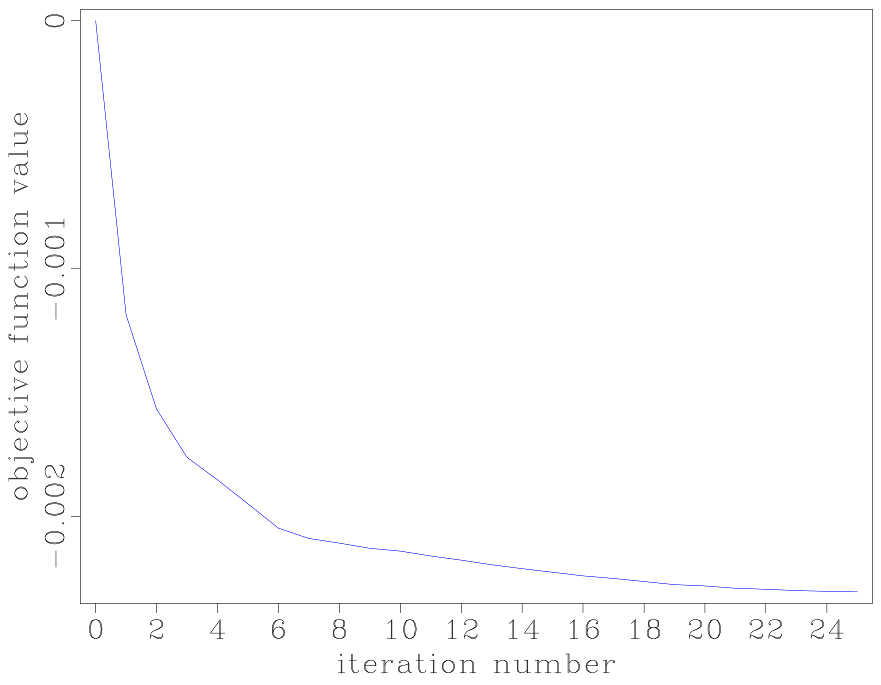


(a)

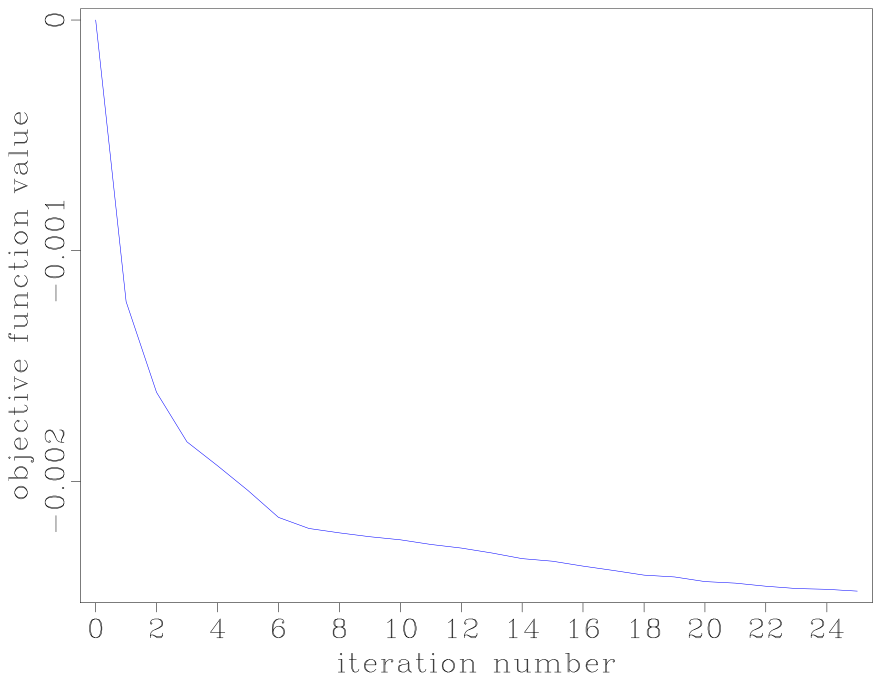


(b)

Figure 2.25: The objective functions from the Hessian inversions using the single canyon perturbation model. Note, the values are negative because a conjugate gradient (CG) solver was used instead of a CG least-squares solver. a) Gauss-Newton Hessian. b) Full Hessian. [**CR**] $\boxed{\text{chapter2/. objfunc-single-gn,objfunc-single-full}}$



(a)



(b)

Figure 2.26: The nearly identical objective functions from the Hessian inversions using the double canyon perturbation model. Note, the values are negative because a conjugate gradient (CG) solver was used instead of a CG least-squares solver. a) Gauss-Newton Hessian. b) Full Hessian. [**CR**] $\boxed{\text{chapter2/. objfunc-double-gn,objfunc-double-full}}$

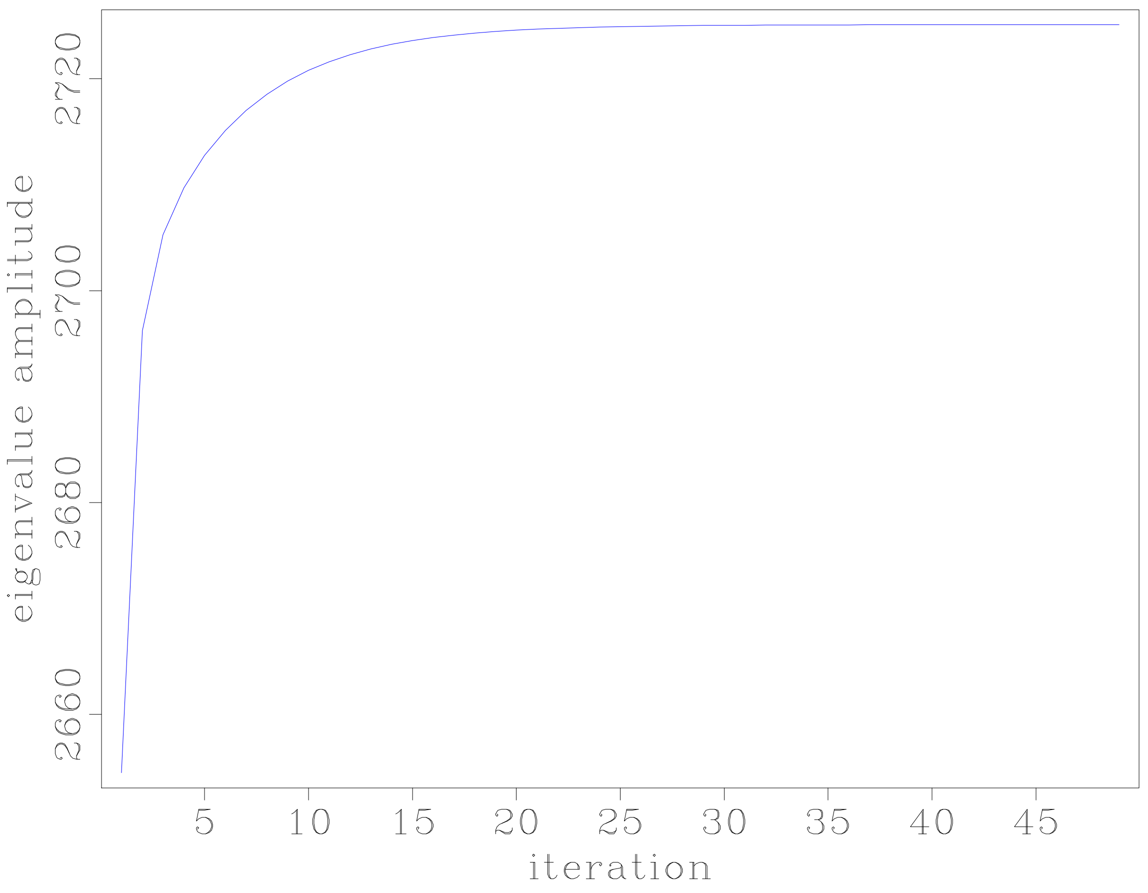


Figure 2.27: The power iteration curve showing the progressing approximation of the maximum absolute value eigenvalue of the full Hessian operator used on the single canyon perturbation model. [**CR**] $\boxed{\text{chapter2/. powerit1}}$

the first one derived is the magnitude of the most negative eigenvalue. I experimented with this method, but found the benefits of this effort to be minimal, and at notable computational cost. Figure 2.27 shows that in practice at least 25 iterations (and so $\sim$ 25 forward full Hessian operator applications) were necessary for each of the two power iteration searches. Once these searches were complete and a proper shift was found, I found that the results of using this Levenberg-Marquardt shift were almost imperceptible from the Gauss-Newton results. Furthermore, since the Hessian operator is model-dependent (and so changes with each outer loop iteration of FWI), these power iteration steps would need to be performed each time the Newton system was inverted.

CONCLUSIONS

Level set concepts can be combined with the FWI objective function to create a shape optimization scheme that allows an elegant way to address the problem of finding an optimal salt body boundary. My demonstrations on simple 2D examples illustrate the relationship between the FWI and implicit surface search directions, and the effect that updating this new model space ultimately has on the velocity model. When I investigate the use of the inverse Hessian to refine the search direction, I find that the Gauss-Newton Hessian approximation is sufficient to improve convergence. However, the theoretically more accurate full Hessian gives mixed results which depend on the model. Robust inversion can only be assured by performing more computation (power iterations) to find an optimal correction for the diagonal elements of the operator. For this reason, I find that the impracticalities of maintaining stability in the full Hessian inversion outweigh the potential benefits from using it. While the Gauss-Newton Hessian is less accurate than the full Hessian, its inversion is stable, and at far less cost.

Chapter 3

Radial basis functions for model sparsity

In the previous chapter, I reviewed the advantages of using the inverse Hessian to improve the search direction used in the FWI problem. In this chapter, I first discuss how re-parameterizing the implicit surface to a sparse domain with Radial Basis Functions (RBFs) may improve the theoretical convergence rate of the Hessian inversion. Next, I demonstrate how RBFs can give a satisfactory representation of a dense implicit surface when suitable parameters are chosen. Last, I give a comparison of level set inversions showcasing the improved convergence rate that is gained by using RBFs.

MOTIVATION

Any uniform increase in the 3D model size increases the number of model parameters by $O(n_x n_y n_z)$. When using a Newton method as described in equation 2.17, we make use of a Hessian that (in matrix form) has $(n_x n_y n_z)^2$ elements. As the model expands to 3D, this quickly becomes intractable to store in memory or even on disk. In practice, we overwhelmingly prefer to solve the inverse Hessian system (equation 2.17) using the conjugate gradient method, which only requires forward applications of the Hessian operator (in our case, the Gauss-Newton Hessian). Regardless, the upper limit of the number of iterations needed to converge fully using the conjugate-gradient method is inversely proportional to the number of model parameters (Aster et al., 2013). As such, our desire to improve the rate of convergence when solving the Newton system motivates us to reduce the number of model parameters in some manner.

For 3D seismic wave propagation, the coarseness of spatial sampling is based on constraints imposed by the numerical dispersion and numerical stability inherent in our finite differencing scheme. For the purpose of wave modeling, our spatial resolution will generally be higher than the features we can robustly resolve from the data. We can typically use uniform down-sampling methods on the update gradient to create a smaller model space without significantly affecting our end results (Cox and Verschuur, 2001). One of the disadvantages of using down-sampling to reduce the model space is that it uniformly reduces resolution. This runs counter to our interest in having high resolution in specific areas of interest, like the salt boundaries. For this reason, any application of down-sampling usually gives less than satisfactory results for inversion resolution.

In the level set problem, we are most interested in areas around and within the salt boundaries. The implicit surface tracking our salt boundary doesn't need to be represented by a fine grid across the whole domain like the one used for wave propagation. However, aggressive down-sampling schemes will quickly start to deteriorate the information that we wish to keep. What is needed is a method that allows the flexibility of spatially varying resolution.

RADIAL BASIS FUNCTION IMPLEMENTATION

One way to achieve model space reduction is with a basis function that lets us use fewer model parameters to describe a larger spatial area. Radial basis functions are a simple and effective kernel to use. We can separately scale and then sum an aggregation of RBFs to approximate a dense implicit surface, with the approximation accuracy related to the shape of the RBF kernel and the number of RBF kernels used. Kadu et al. (2017a) propose an implementation like this which replaces a dense Cartesian grid parametrization of the implicit surface $\boldsymbol{\phi}$ with a surface described as an aggregate of evenly spaced RBFs:

$$\boldsymbol{\phi}(\boldsymbol{\lambda};\epsilon) = \sum_{i}^{N_\lambda} \lambda_i \exp^{-(\epsilon \mathbf{r}_i)^2}. \tag{3.1}$$

In this formulation, $\boldsymbol{\lambda}$ is the new (sparse) model parameter vector, N_λ is the length of $\boldsymbol{\lambda}$, $\mathbf{r}_i$ is an array of radial distance from the ith RBF center with size $N_z \cdot N_x \cdot N_y$ for the 3D spatial case, and ϵ controls the sharpness of the RBF taper (constant for all i). In their approach, Kadu et al. (2017a) chose a uniform distance between each RBF center location beforehand based on the resolution they desired.

However, there are only a few areas where one really wants high resolution (namely, the salt edge where we expect boundary movement). The resolution achieved with the sparse parametrization is primarily based on how many RBFs are used to describe a particular area (RBF density). Because the salt evolves from an initial boundary, the further from this initial boundary a model region is, the less likely that it will be updated, which means low RBF density can be justified. On the other hand, regions close to the current salt boundary are more likely to update, justifying higher RBF density. Therefore, I introduce the idea of spatially varying the density of the RBF centers to represent the implicit surface in a sparse fashion (as shown in Figure 3.1). This allows clustering the center locations of the RBFs in areas I expect to see updating occur, while using a lower density in regions where I don't expect updating (see Figure 3.2). This is a more efficient way to distribute RBF positions, resulting in fewer RBF parameters needed to attain high resolution around the salt boundary than if I used the regular RBF spacing described in Kadu et al. (2017a).

For the new representation of $\boldsymbol{\phi}$ described in equation 3.1, the operator $\mathbf{D}$ must be modified to account for the additional linear transformation:

$$\begin{aligned}\mathbf{D} &= \begin{bmatrix} \frac{\partial \mathbf{m}(\boldsymbol{\phi_o},\mathbf{b_o})}{\partial \boldsymbol{\phi}} \frac{\partial \boldsymbol{\phi}}{\partial \boldsymbol{\lambda}} & \frac{\partial \mathbf{m}(\boldsymbol{\phi_o},\mathbf{b_o})}{\partial \mathbf{b}} \end{bmatrix} \\ &= \begin{bmatrix} \widetilde{\delta}(\boldsymbol{\phi_o})(\mathbf{c_s} - \mathbf{b}) \exp^{-(\epsilon \mathbf{r})^2} & \mathbf{I} - \widetilde{H}(\boldsymbol{\phi_o}) \end{bmatrix}. \end{aligned} \tag{3.2}$$

In this formulation, $\widetilde{H}$ is the Heaviside approximation (equation 2.11), $\mathbf{I}$ is the identity matrix, $\widetilde{\delta}$ is the derivative of $\widetilde{H}$, $\mathbf{r}$ is a tensor of size $N_z \cdot N_x \cdot N_y \cdot N_\lambda$ for the 3D spatial case, $\mathbf{b}$ is the background velocity ($\mathbf{b_0}$ is fixed), $\boldsymbol{\phi}$ is the implicit surface ($\boldsymbol{\phi_0}$ is fixed), and $\mathbf{c_s}$ is the constant salt velocity. Further, the model space has also changed:

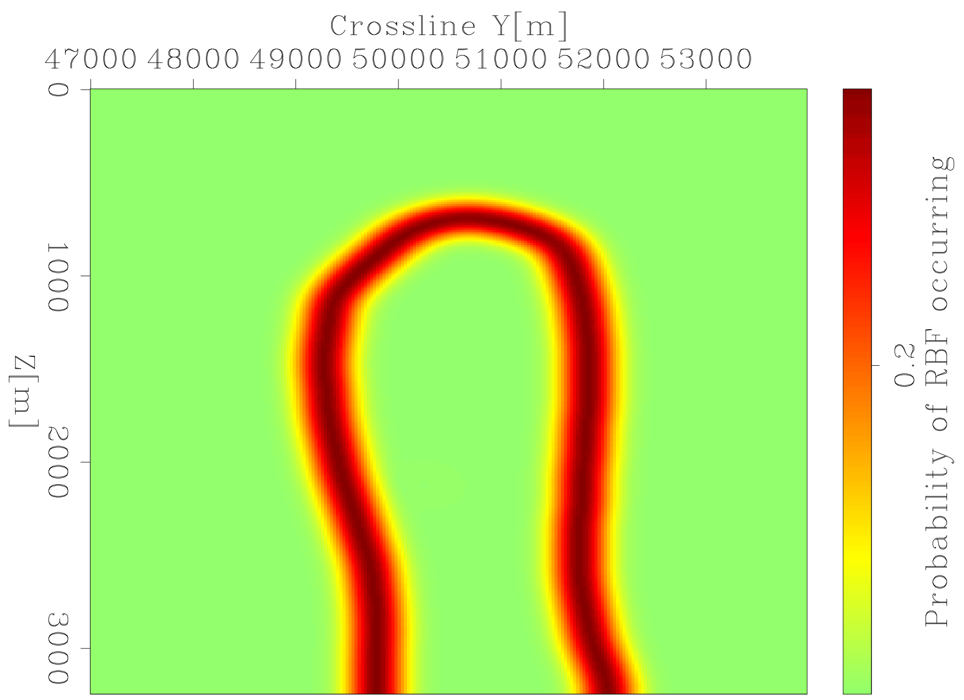


Figure 3.1: The probability distribution used to randomly position the radial basis function centers in Figure 3.2. [**ER**] chapter3/. centers-dist

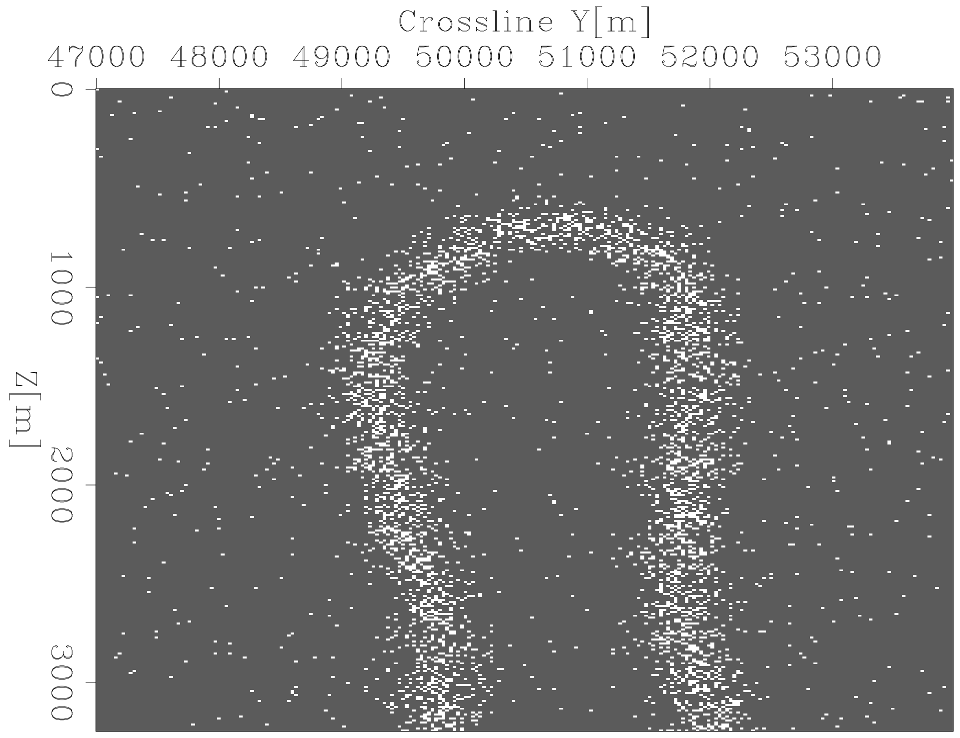


Figure 3.2: Center positions for radial basis functions used to construct the implicit surface. [**ER**] chapter3/. centers

$$\triangle \mathbf{p} = \begin{bmatrix} \triangle \boldsymbol{\lambda} \\ \triangle \mathbf{b} \end{bmatrix}.$$

When I apply the operator $\mathbf{D}$ or $\mathbf{D^T}$, I consider the locations of the RBF centers to be fixed throughout the inversion (for example, as shown in Figure 3.2).

Computational considerations

The forward application of the $\mathbf{D}$ operator derived earlier now includes a $\exp^{-(\epsilon \mathbf{r})^2}$ term. This means that for each element λ_i in the model vector, I scale and sum a Gaussian function to the aggregated surface, $\boldsymbol{\phi}$. If I compute $\exp^{-(\epsilon \mathbf{r})^2}$ over the full model, then the $\mathbf{D}$ operator becomes expensive to apply on a large 3D spatial domain, since the algorithm would loop over the full model space for each RBF. One observation that I leverage is that the value of the radial basis function decreases significantly at high values of r. Furthermore, I scale and sum the same Gaussian each time (ϵ is fixed). Based on this, I pre-compute the radial basis function $\exp^{-(\epsilon \mathbf{r})^2}$ just once over a region where its value is actually significant. Naturally, the size of this region is based on the taper of the RBF, which is governed by ϵ. By choosing ϵ well, and then pre-computing the corresponding Gaussian function over a limited region, the RBF summation computation is greatly simplified, and increases the speed of applying $\mathbf{D}$ or $\mathbf{D^T}$ significantly.

EXAMPLES OF RBF FITTING

In order to show how RBFs can accurately represent a salt body with far fewer parameters than a dense representation, I demonstrate their use on a Gulf of Mexico velocity model provided by Shell. I choose a section of the velocity model that has a notable salt protrusion in it as an example (Figure 3.3). Beginning with this model, I build a probability density map that favors placing RBF centers near the original picked boundary (Figure 3.1), with less likelihood further from that boundary. From this, I generate random RBF positions (Figure 3.2). Using these RBF centers, I then perform a non-linear conjugate gradient inversion to find the proper weighting of the

RBF kernels in order to best fit the starting model salt shape (Figure 3.4(b)), which is built from an initial implicit surface (Figure 3.4(a)). The inversion finds sparse parameters that create a new implicit surface (Figure 3.5(a)), which looks different from the one used to build the fitting model (Figure 3.4(a)). However, when I apply the Heaviside function to either of these to create the salt, we can see that the sparse parameters create a model with a good fit (compare Figures 3.4(b) and 3.5(b)). The inversion converges relatively quickly (Figure 3.6), but naturally has some unresolved residual since the RBF parametrization is sparse and cannot match the dense salt model guess perfectly (see Figure 3.7(a)).

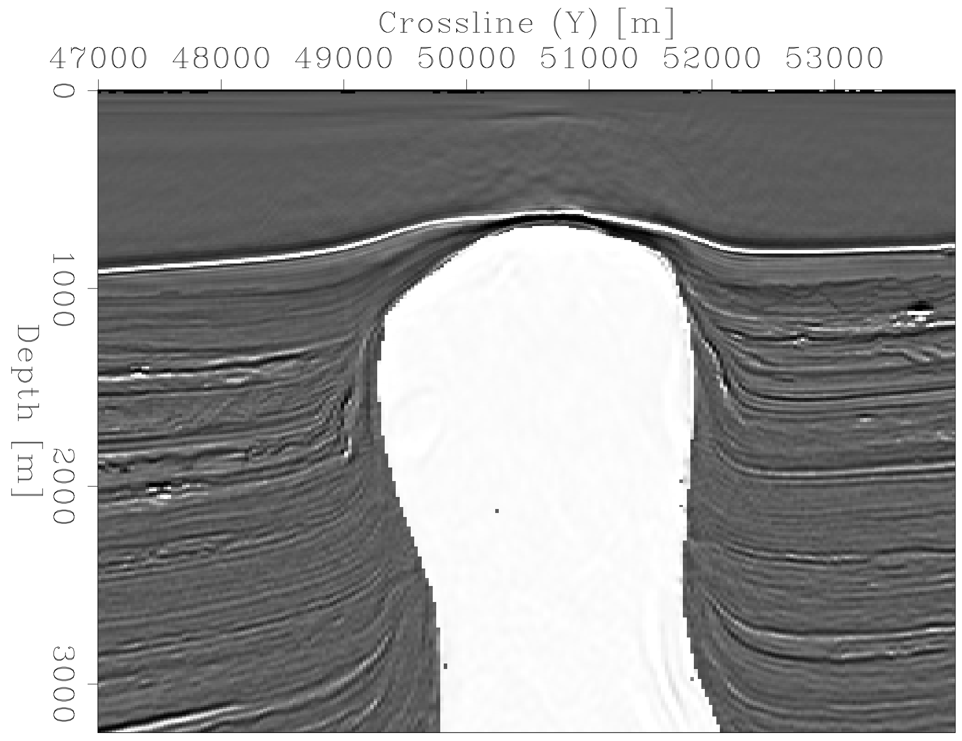
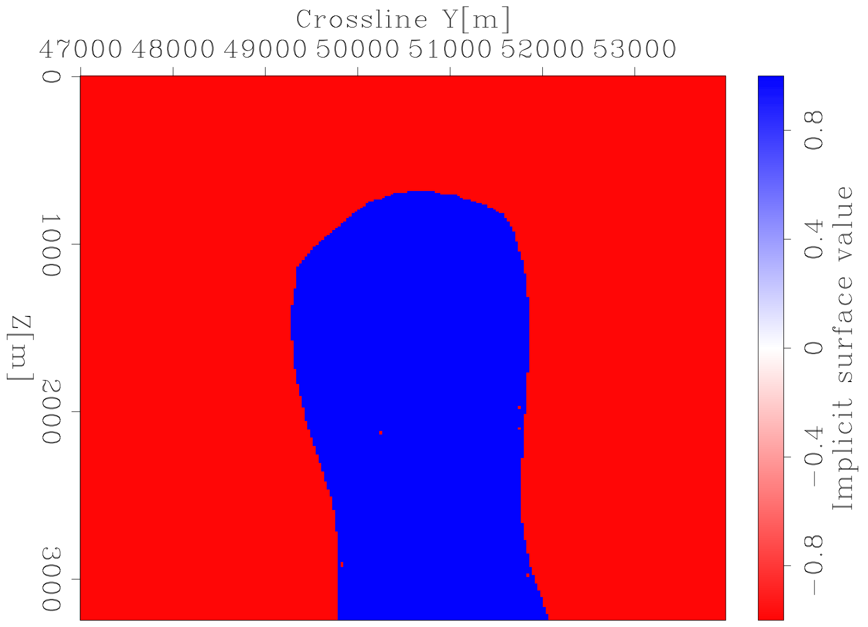
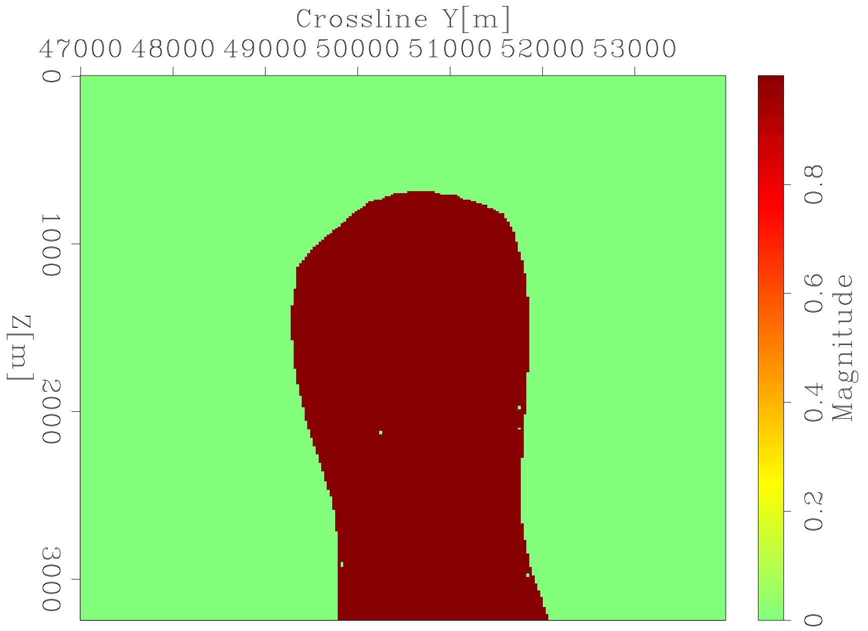


Figure 3.3: Salt model used by Shell (white) overlaid on the corresponding RTM image. [**ER**] chapter3/. both

Figure 3.7(a) shows that the matching model and the resulting inverted model after Heaviside function application are quite similar. However, if we choose ϵ and the corresponding RBF footprint poorly, we wont be able to represent the original salt model as well as we could. When ϵ is too high, the RBF decays quickly, resulting in a model that is less smooth. Alternatively, when ϵ is too low, the RBF decays slowly and creates a model that is too smooth. Figure 3.7(b) shows a case where these parameters were chosen poorly, while Figure 3.7(a) does a much better job in

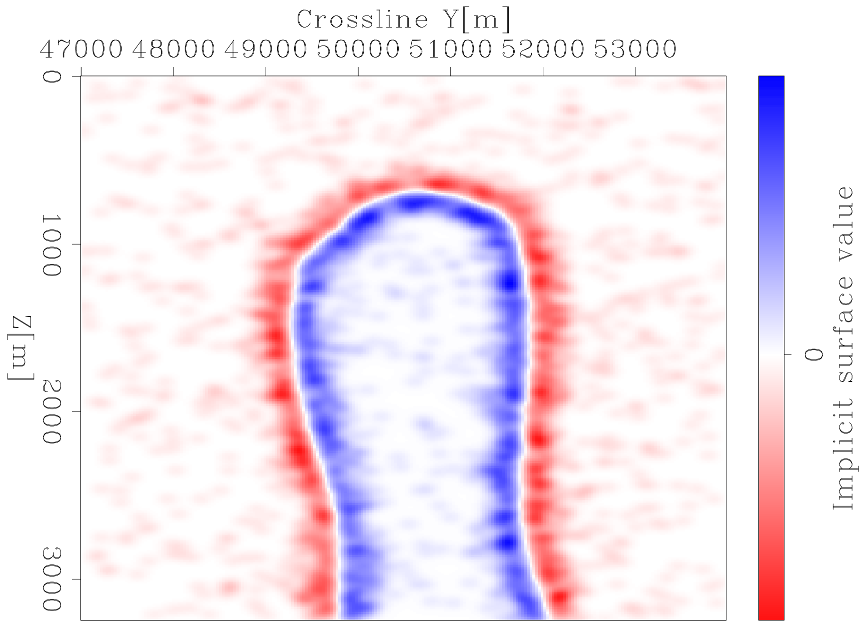


(a)

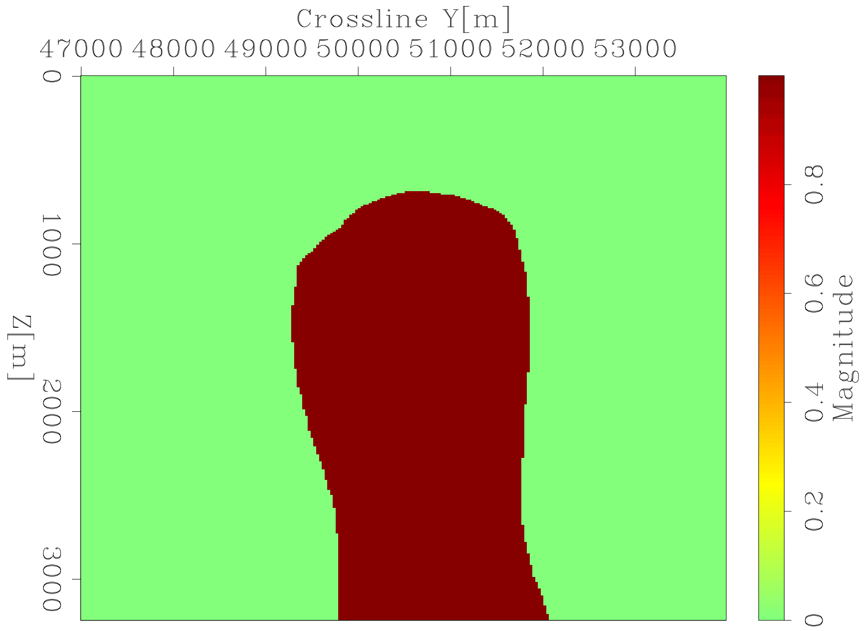


(b)

Figure 3.4: a) Implicit surface $\boldsymbol{\phi}_\mathrm{o}$ that creates the salt body shape (b) that my inversion tries to match. b) is the result of applying the Heaviside function to $\boldsymbol{\phi}_\mathrm{o}$. chapter3/. matching-phi,matching-phi-Heavi



(a)



(b)

Figure 3.5: a) Implicit surface $\boldsymbol{\phi}_{\text{final}}$ created from final inverted RBF parameters ($\boldsymbol{\phi}_{\text{final}} = \mathbf{D}\boldsymbol{\lambda}_{\text{final}}$); b) the result of applying the Heaviside function to $\boldsymbol{\phi}_{\text{final}}$. $\boxed{\text{chapter3/. resulting-phi,resulting-phi-Heavi}}$

both the salt center region as well as the boundary.

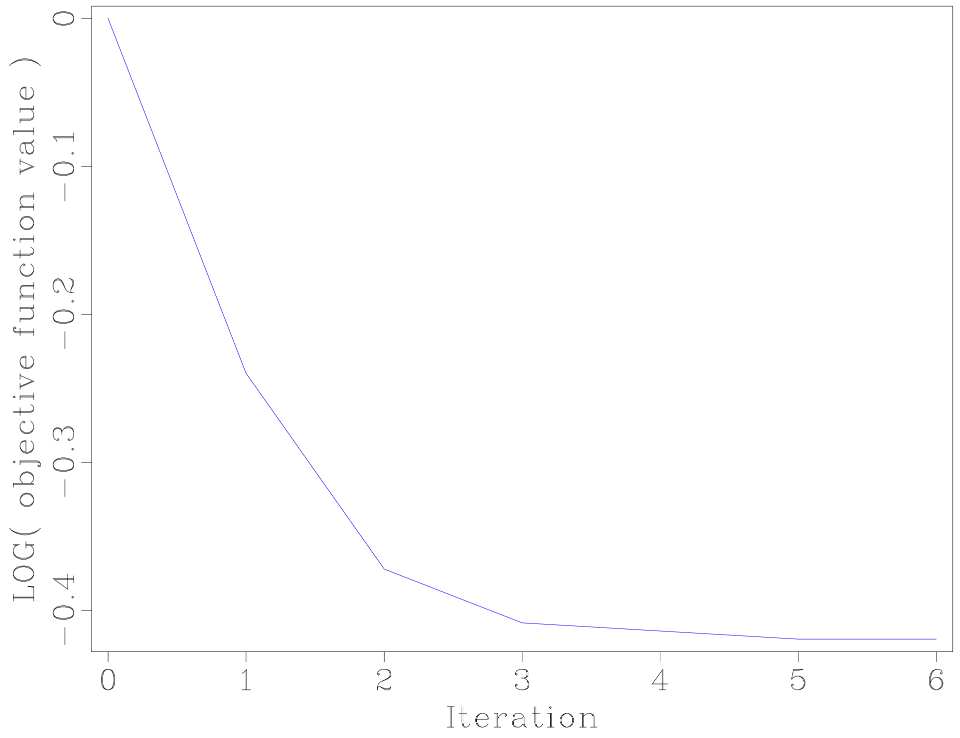
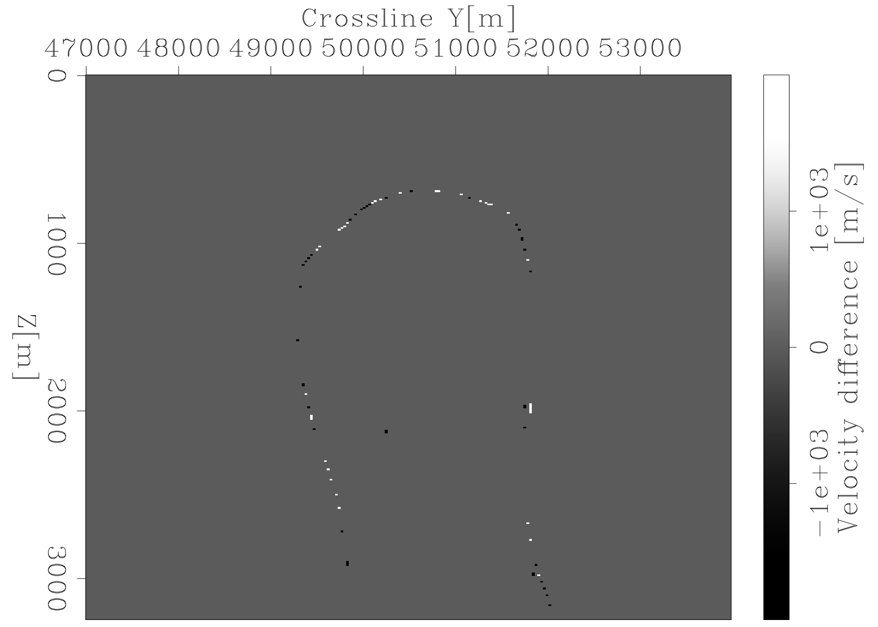


Figure 3.6: Log of normalized objective function from the non-linear inversion used to find the RBF parameters used in Figures 3.5(a), 3.5(b) and 3.7(a). [**ER**] chapter3/. objfunc

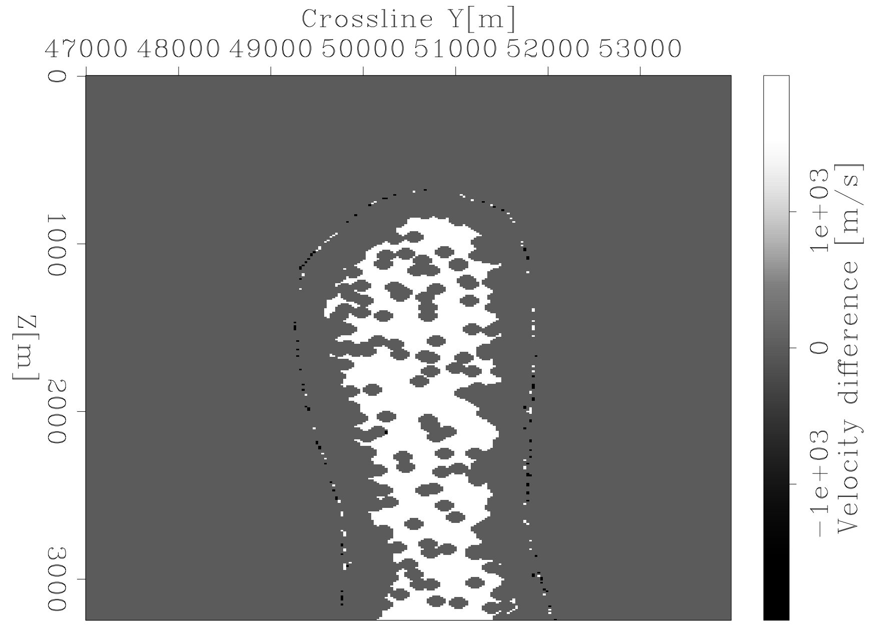
COMPARISON OF SPARSE (RBF) AND DENSE MODEL INVERSIONS

In order to illustrate the improved rate of convergence gained by using a sparse RBF model, I demonstrate on a 2D synthetic inversion example. The goal here is to show how RBFs affect the greater shape optimization problem by comparing the inversion convergence rates between a sparse RBF parametrization and a dense non-RBF parametrization.

I choose a 'true' model similar to Figure 3.3 that I wish to invert for (Figure 3.8) that has an inclusion close to the edge. This model is chosen to show how the RBF inversion can invert for a more unusual model geometry. I begin with a model containing a much smaller inclusion (Figure 3.9). Just as in chapter two (Algorithm 1), the inversion workflow has an outer loop where I do non-linear modeling to find the



(a)



(b)

Figure 3.7: Differences between original salt model and the resulting model produced by the RBF representation. (a) shows difference of fitted salt model using $\epsilon = 0.25$ value, while (b) shows difference of fitted salt model using $\epsilon = 2.25$ value. Both cases used 98% fewer model parameters than the original full-grid scheme, and both used the same RBF positions. Background velocity is 2.5 km/s and salt velocity is 4.5 km/s. [**ER**] chapter3/. rbfinv-diff-full,rbfinv-diff-sparse

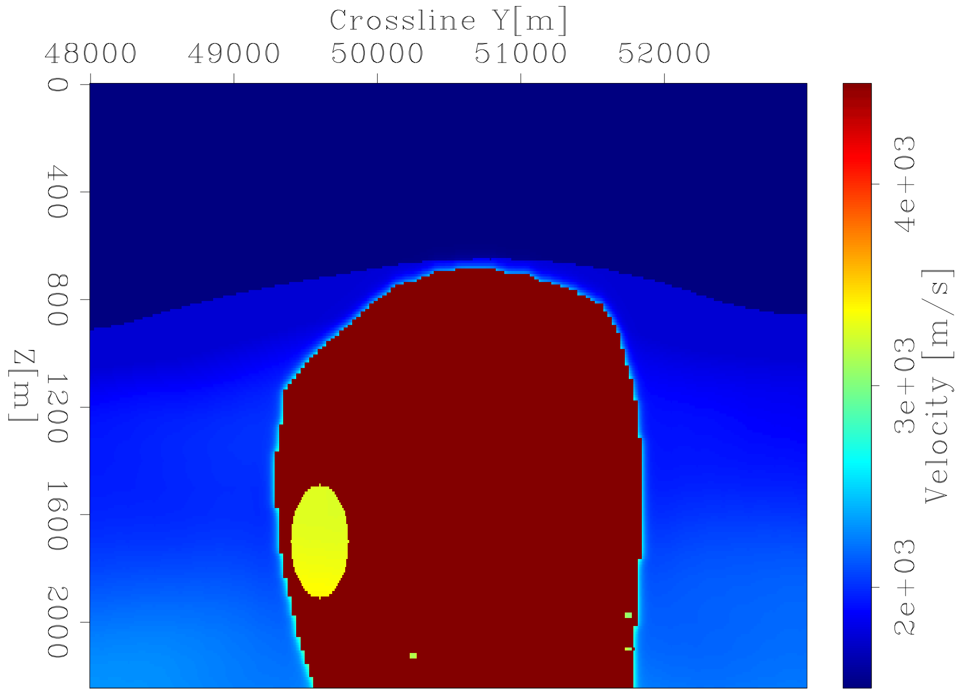


Figure 3.8: True velocity model that was used to synthetically generate the ‘observed’ data. [**ER**] chapter3/. true-model-rbf-inversion

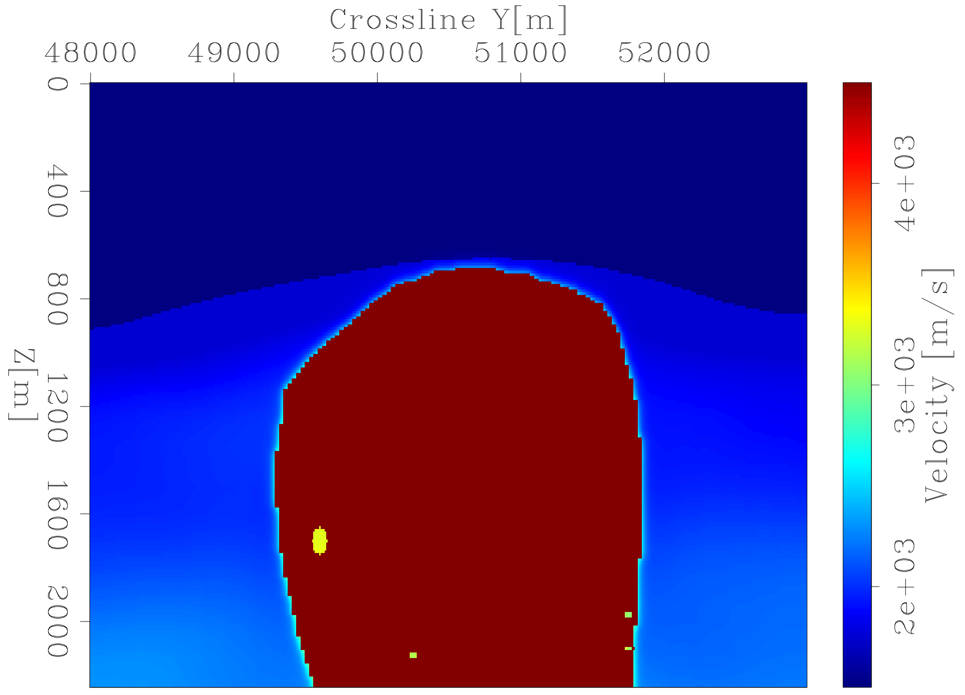


Figure 3.9: Starting velocity model. Note the inclusion is much smaller than in the true model (Figure 3.8). [**ER**] chapter3/. start-model-rbf-inversion

residual and use the adjoint Born operator to find the gradient. However, following this I now have an inner loop using iterative methods to invert the Hessian and find the search direction from the gradient (solving equation 2.17), before performing the linesearch. In these examples, the inner loop uses a Gauss-Newton Hessian, and I compare using a dense model (no RBF parametrization) to using a sparse model (with RBF parametrization). The RBF model has only 7% of the model points that the dense model has. I perform the same number of iterations (20) for each inner-loop linear inversion of the Gauss-Newton Hessian using a conjugate gradient solver. For the first Hessian inversion, the sparse RBF parameterization (Figure 3.10) converges faster than the dense model example (Figure 3.11). Note that the objective function curves become negative since I use a conjugate gradient (CG) solver based on equation 3.3:

$$\min \psi(\triangle \mathbf{m}) = \frac{1}{2}\triangle \mathbf{m}^{\mathbf{T}}\mathbf{H}\triangle \mathbf{m} - \mathbf{g}^{\mathbf{T}}\triangle \mathbf{m}, \tag{3.3}$$

instead of a conjugate gradient least-squares (CGLS) solver (based on equation 3.4):

$$\min \psi(\triangle \mathbf{m}) = \frac{1}{2}||\mathbf{H}\triangle \mathbf{m} - \mathbf{g}||_2^2. \tag{3.4}$$

Because the Hessian is a symmetric operator, I can take advantage of using the CG solver instead of the more expensive CGLS solver. However, the residual is not squared as in CGLS, so the objective function can take negative values.

I find that after 14 outer loop (non-linear) iterations the sparse (RBF) inversion converges, while the dense (non-RBF) inversion objective function is still descending after 40 iterations (Figure 3.12). The normalized model norm for the sparse inversion also reaches a lower value (Figure 3.13), while the non-RBF inversion actually increases instead. By representing the dense model sparsely with RBFs, I reduce the number of parameters as well as create a smoother equivalent update in the dense model space (with smoothness based on the ϵ used). In this sense, the RBFs act somewhat like a regularization. Both these factors contribute to the improved convergence rate that I find when using RBFs in the inversion. When comparing the inverted model results of each parametrization, one can clearly see that the RBF approach (Figure 3.14) provides a superior result to the dense model space parametrization (Figure 3.15).

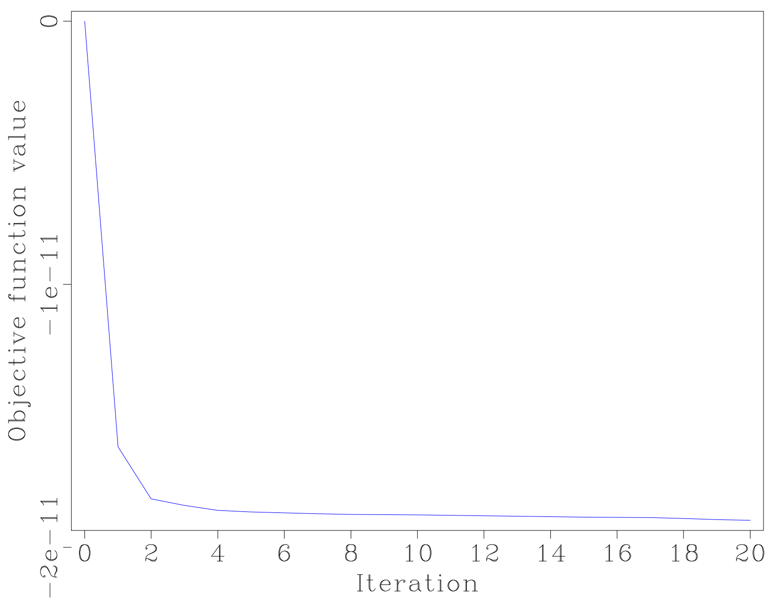


Figure 3.10: Objective function from the first inner-loop Gauss Newton inversion of the sparse (RBF) model system. [**CR**] $\boxed{\text{chapter3/. GNinversion-rbf-objfunc-0}}$

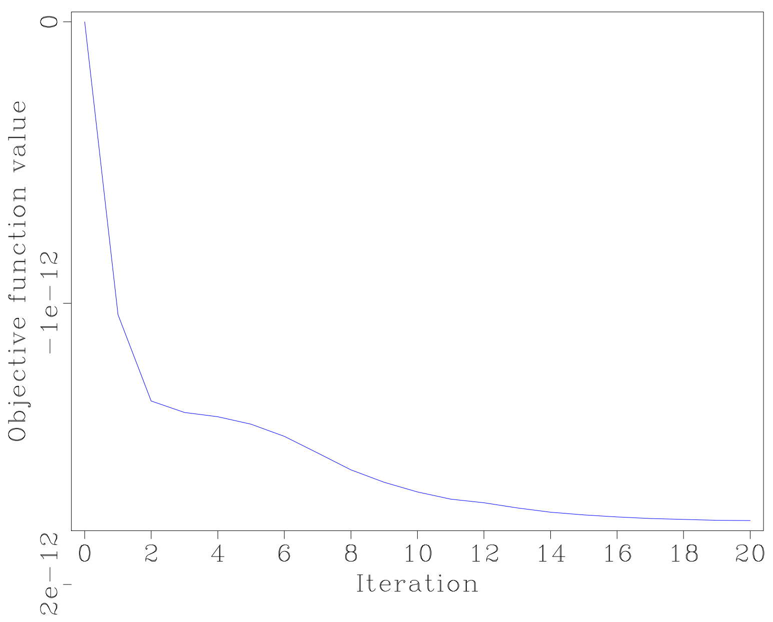


Figure 3.11: Objective function from the first inner-loop Gauss Newton inversion of the dense (non-RBF) model system. [**CR**] $\boxed{\text{chapter3/. GNinversion-norbf-objfunc-0}}$

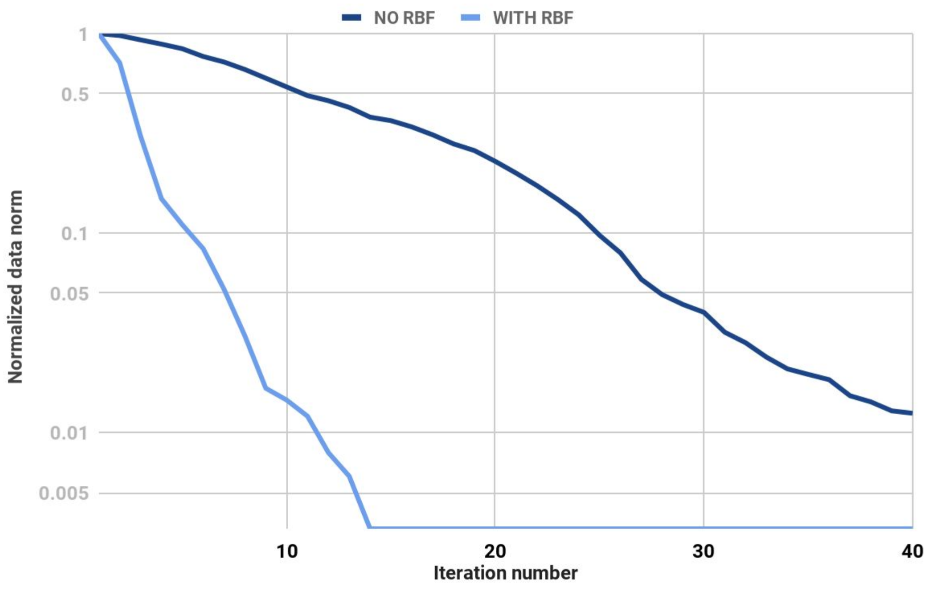


Figure 3.12: Objective function (data norm) comparison of the outer-loop inversion for RBF and non-RBF parameterized model approaches. [**CR**] $\boxed{\text{chapter3/. RBF-vs-noRBF-dataNorm}}$

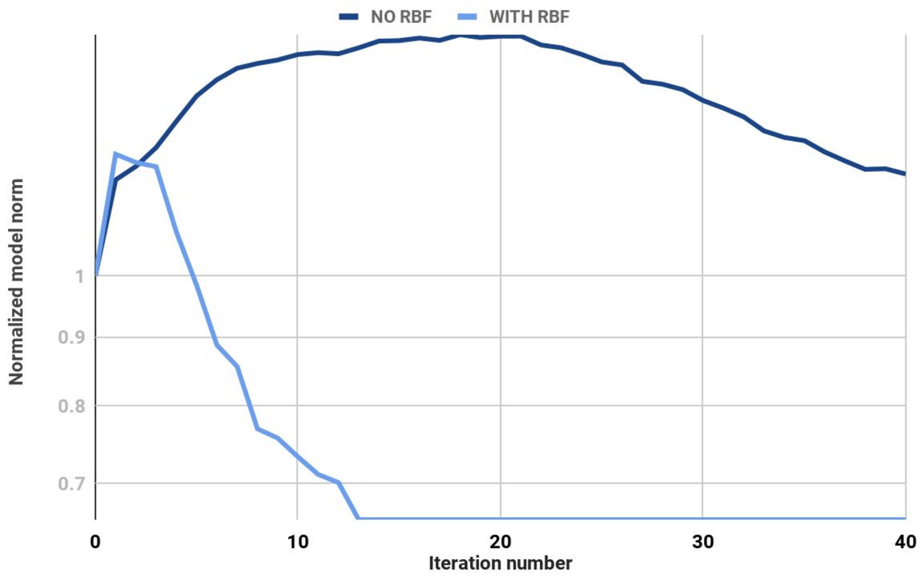


Figure 3.13: Model norm comparison of the outer-loop inversion for RBF and non-RBF parameterized model approaches. [**CR**] $\boxed{\text{chapter3/. RBF-vs-noRBF-modelNorm}}$

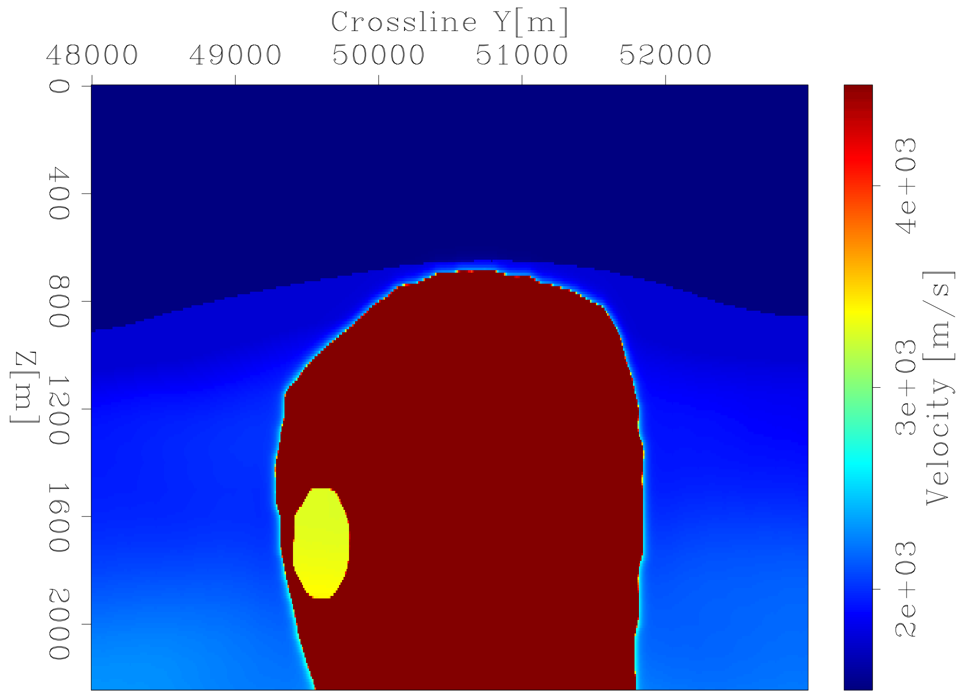


Figure 3.14: Velocity model after 30 outer loop (non-linear) iterations using RBF parametrization. chapter3/. model-30-RBF-inversion

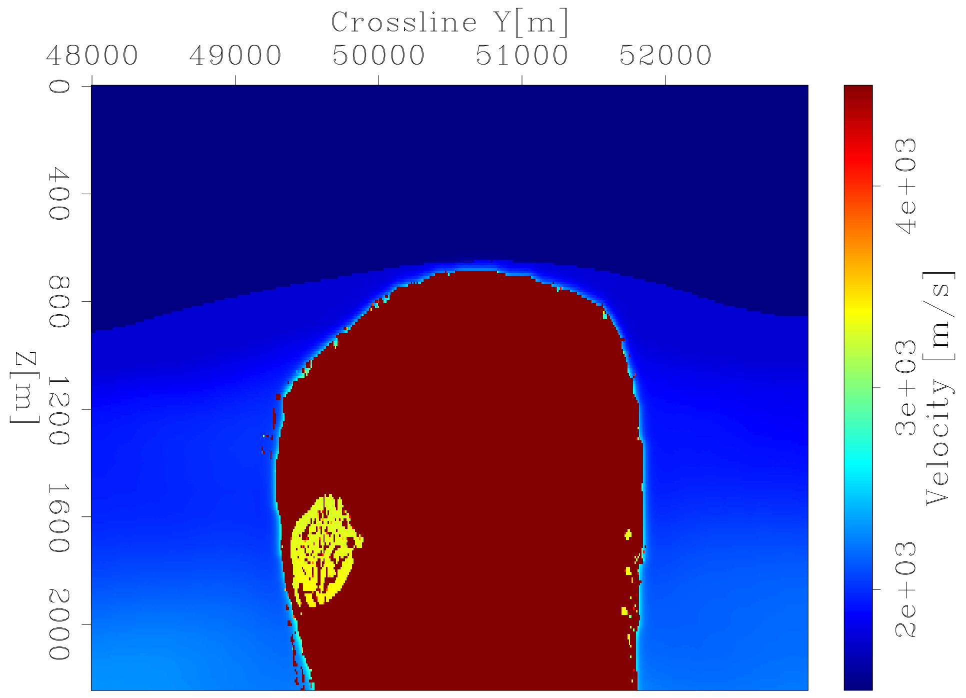


Figure 3.15: Velocity model after 30 outer loop (non-linear) iterations without RBF parametrization. chapter3/. model-30-noRBF-inversion

CONCLUSIONS

The speed at which one can invert the Hessian system and find the search direction is sensitive to the number of parameters in the model, and using 3D spatial models requires a large number of model parameters for wave propagation. However, I can sparsely represent this dense model using radial basis functions to achieve significant parameter reduction. I show that this representation allows me to accurately depict the dense model, and that steps can be taken to make this transform computationally efficient. Finally, when I compare inversions using a sparse representation (RBFs) versus a dense representation, I find that the sparse model inversion provides a better outcome and a faster convergence rate.

Chapter 4

Interpreter guidance for inclusion discovery

In this chapter I explain how one can modify the implementation of the level set inversion so that outside input can elegantly guide the inversion by means of expanding the footprint of the gradient update and by intelligently initializing the implicit surface. This improves the overall convergence rate, which is important because the operators used to find the search direction are computationally expensive. I finish by comparing inversion examples that showcase how these types of guidance can improve the convergence rate and allow for model updating that the unmodified level set algorithm is unable to replicate.

MOTIVATION

An advantage of the level set framework is that the implicit surface can be deformed at positions that are not on the current boundary, even to the extent that it 'punctures' the zero level set and creates a 'donut hole' or other topology. This can be useful for modifying salt bodies, since we may begin with a solid salt body, and then later deform the implicit surface such that the salt has inclusions inside it.

However, an underlying assumption of the gradient derived earlier is that the salt model only changes at the boundary of the original shape. This is because the salt is defined using a Heaviside function approximation, and so the gradient contains a $\delta(\phi_0)$ term that (in a practical sense) updates at the boundary only. If we only update along the current boundary in this manner, we stifle the possibility of topology changes, such as the discovery of inclusions that exist away from the current boundary. This is an inherent limitation of the theory I have derived thus far.

Furthermore, I have yet to discuss any methodology for initializing the implicit surface prior to inversion other than stipulating that the zero crossing position correlates to the boundary of the salt body with which we wish to begin. As long as the interior salt region is greater than zero, the Heaviside function indicates it as salt. This means that beyond the requirement of having the proper sign, there is flexibility in how one initializes that surface. I leverage this flexibility to create regions of preferential updating sensitivity.

MODIFICATIONS TO GRADIENT

Because the support of the gradient I previously derived is a subset of the actual objective function support (which is the entire model domain), I can expand the support of the masking term to preserve regions inside the current salt boundaries (possible inclusion regions) without affecting our ability to minimize our objective function. This means I can modify the masking term in our $\mathbf{D}$ operator to allow for updating outside of the boundary in regions selected by seismic interpreter guidance. I represent this modification with a new term $\hat{\boldsymbol{\delta}}(\boldsymbol{\phi_0}, \mathbf{G})$, which takes into account the interpreter guidance $\mathbf{G}$:

$$\mathbf{D} = \left[\hat{\boldsymbol{\delta}}(\boldsymbol{\phi_0}, \boldsymbol{G})(\boldsymbol{c}_{\text{salt}} - \boldsymbol{b_0}) \quad \boldsymbol{I} - \hat{\boldsymbol{H}}(\boldsymbol{\phi_0}) \right] .$$

The $\mathbf{G}$ term defines additional regions of our model where one wishes to allow updating to occur, for example, areas where an interpreter suspects inclusions to exist.

INITIALIZING THE IMPLICIT SURFACE USING INTERPRETER GUIDANCE

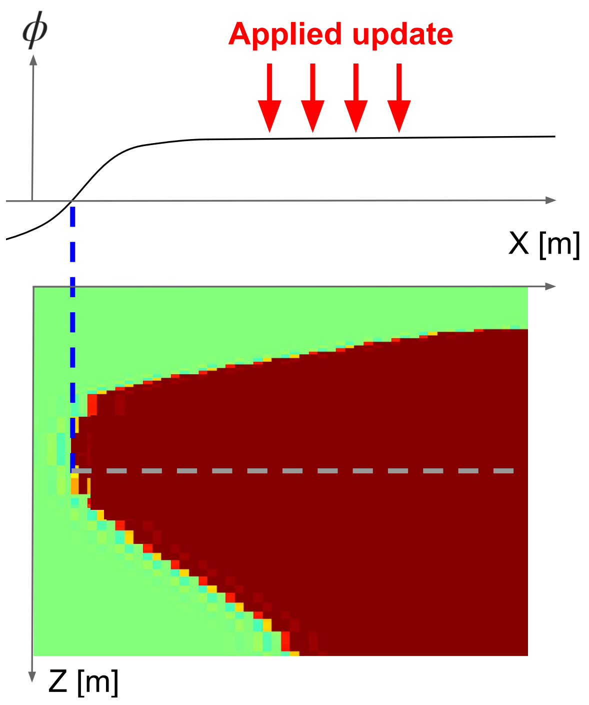
However, even if I expand the footprint of the region where updating is allowed, I can only discover an inclusion if the update perturbs the implicit surface below the zero-level set. This means the initial height of the implicit surface affects sensitivity to updating in different regions. For example, in Figures 4.1(a) and 4.1(b), we can see how an unguided implicit surface can receive an update that suggests an inclusion, but not realize that change in the resulting salt model.

However, I can initialize ϕ as shown in Figure 4.2(a) in order to increase sensitivity in that area of the salt model. This time, the same update is able to create the inclusion in the same iteration (4.2(b)). This is especially important when there are other model regions being updated that have a high impact on the data residual. In the case of Figure 4.1(a), the linesearch that chooses how strongly to apply the update will be most influenced by regions (like the top of salt) to which the objective function is inherently more sensitive, and may never allow for an inclusion update deeper in the model, even after many iterations.

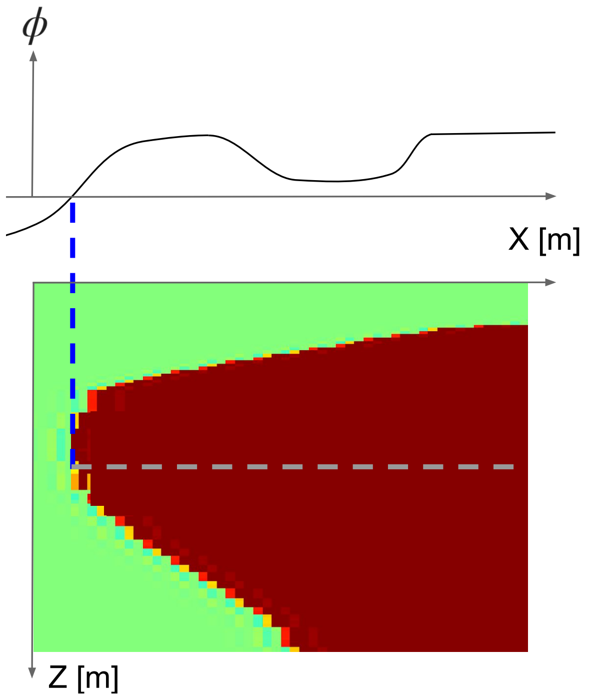
The opportunity here is to set the height of the implicit surface according to how likely we believe that an inclusion is present at that position. This allows us to input a probabilistic mapping of inclusion likelihood into our initialization of ϕ. The same interpreter input G from before can be scaled and used to modify the implicit surface height appropriately. For the Sigsbee model example, Figure 4.3 shows the implicit surface with interpreter guidance applied, and Figure 4.4 shows it without. In Figure 4.3, we can see light-colored regions of the implicit surface that are not present in Figure 4.4, and have a lower value than the rest of the interior salt region. This means updates in this area will be more likely to break past the zero-level set contour and change the value of $\hat{H}(\phi_o)$ to create an inclusion in the model (if the data suggests that). In the case of erroneous interpreter guidance, the gradient may not create an inclusion, or if it does, it may fill it back in during later iterations.

DEMONSTRATION ON A GULF OF MEXICO MODEL

In this 2D synthetic example, I use a model where the inclusion is close enough to the boundary (Figure 4.5(a)) that there is a reasonable chance of the unmodified level set approach finding the inclusion without any of the interpreter guidance methods already described. Beginning with a model that has no inclusion (Figure 4.5(b)), I compare the unmodified method against the partially guided inversion (expanded gradient only) and the fully guided inversion (expanded gradient and implicit surface initialization). For both of the guided approaches, I expand the gradient by modifying

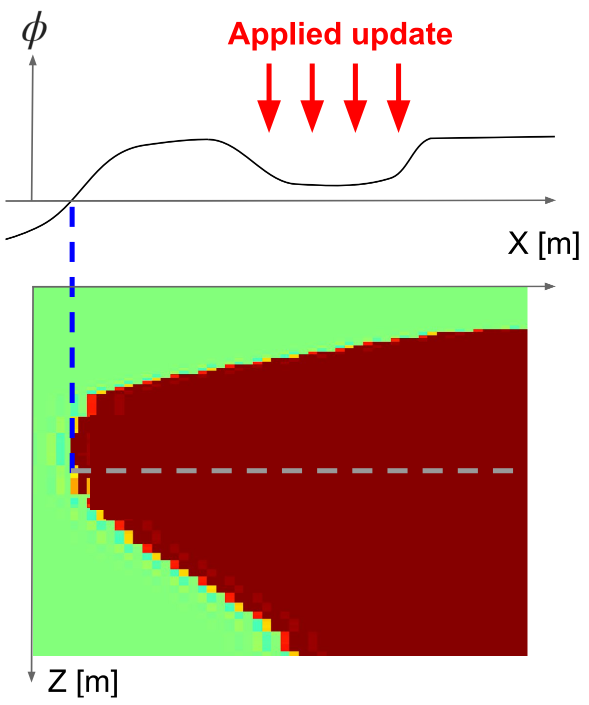


(a)

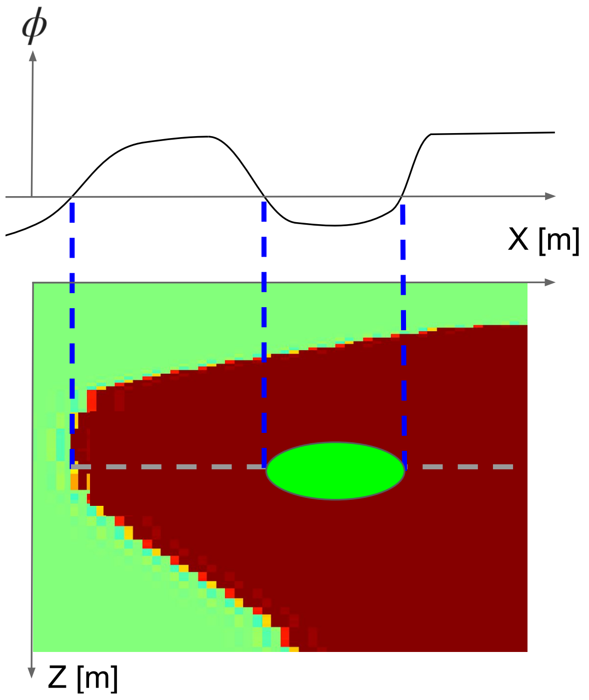


(b)

Figure 4.1: a) Example of salt model before update applied, and b) after update applied to simple ϕ surface with no sensitivity preferences. Note that the update applied in Figure 4.1(a) does not decrease the implicit surface below zero in Figure 4.1(b), resulting in no salt boundary change. [**NR**] $\boxed{\text{chapter4/. inclusion-discovery1a,inclusion-discovery1b}}$



(a)



(b)

Figure 4.2: a) Example of salt model before update applied, and b) after update applied to ϕ surface with sensitivity preferences already incorporated into initial ϕ surface. Note that the update applied in Figure 4.2(a) decreases the implicit surface below zero in Figure 4.2(b), resulting in a salt boundary change. [**NR**] chapter4/. inclusion-discovery2a,inclusion-discovery2b

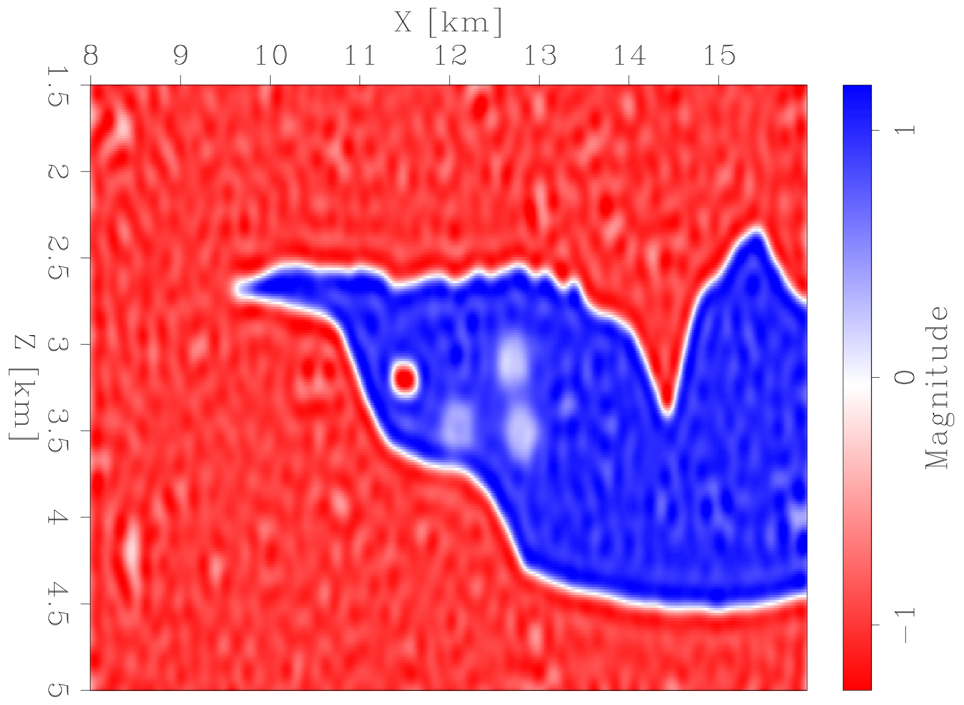


Figure 4.3: Initial implicit surface (guided). Note the three light colored regions where the implicit surface value has been reduced. [**ER**] chapter4/. initialphi-fullyGuided-sigsbee

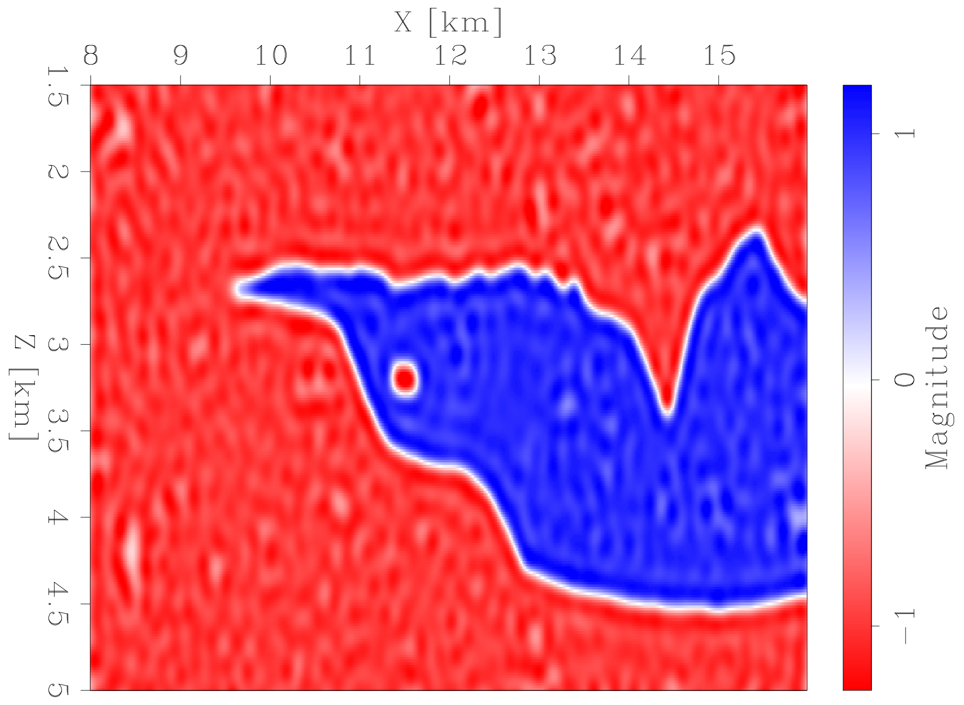


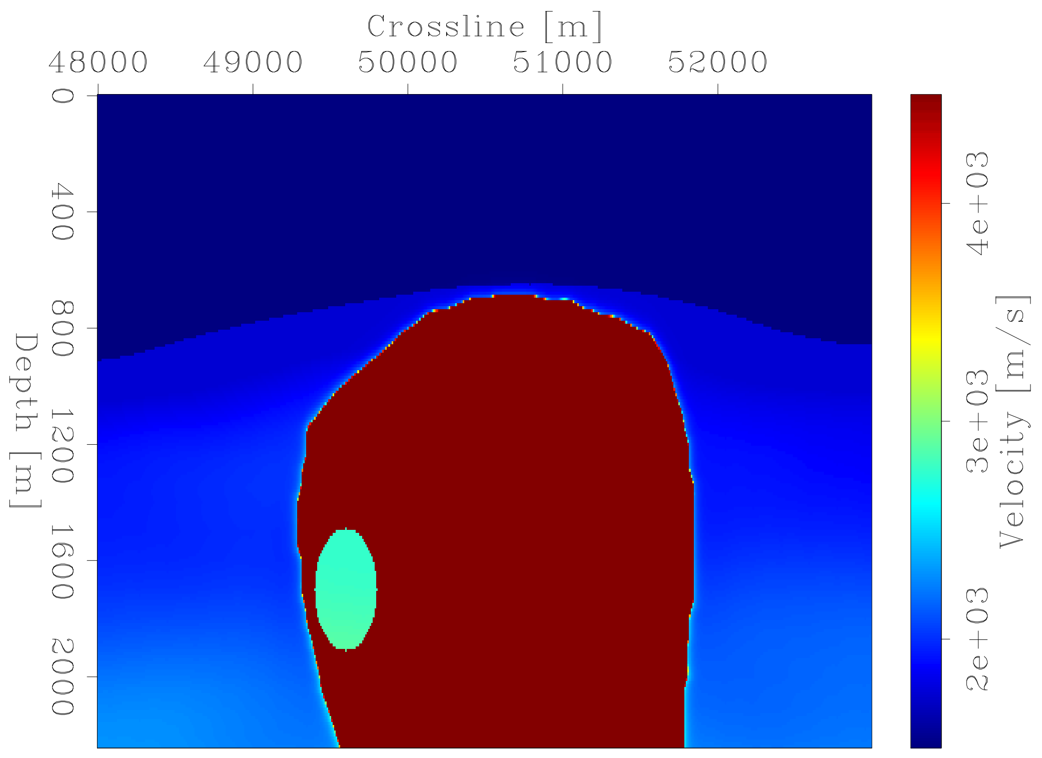
Figure 4.4: Initial implicit surface (unguided). Note the lack of the three light colored regions that are present in Figure 4.3. [**ER**] chapter4/. initialphi-unguided-sigsbee

the masking to allow for updating in the zone where I anticipate that an inclusion exists (Figure 4.6(a)). The unguided approach has no expanded gradient (Figure 4.6(b)). In the fully guided example, I also initialize the implicit surface using the same interpreter guidance so that it has a lower value where I anticipate an inclusion (Figure 4.7(a)), while the unguided and partially guided approaches do not use this guidance for initialization of the implicit surface (Figure 4.7(b)).

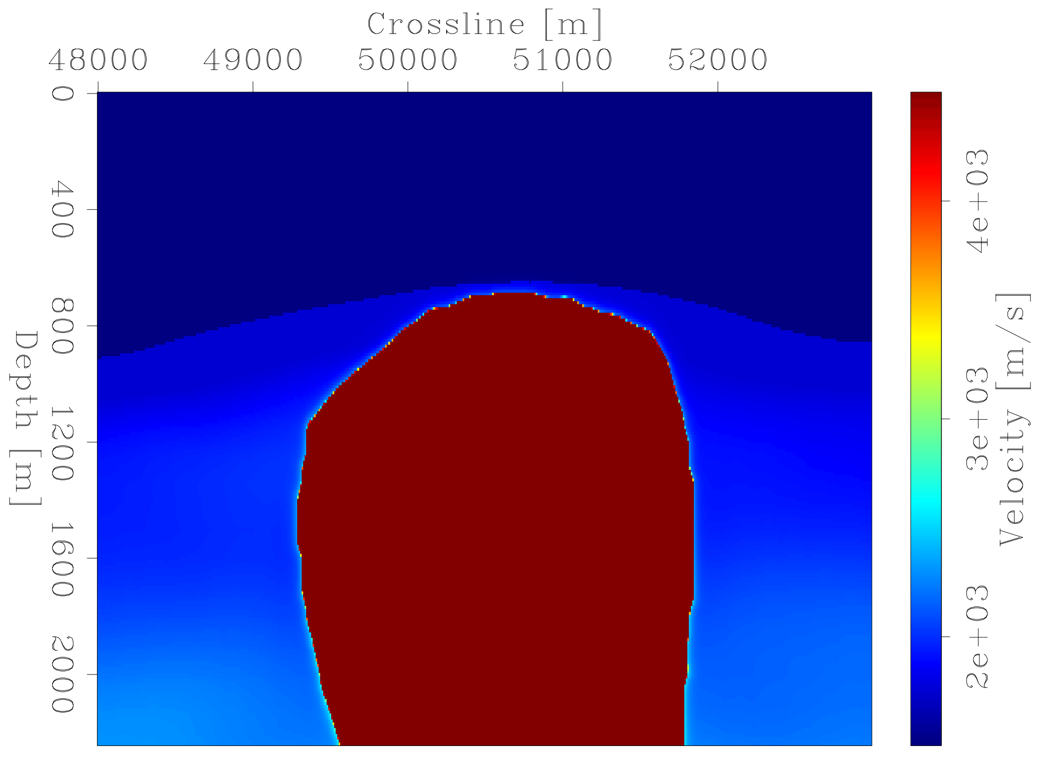
As expected, I find that the guided inversion approaches perform much better (Figures 4.8 and 4.9) in terms of reducing both the model residual and data residual norms. In the partially guided approach, the data guides the inversion to the correct inclusion shape regardless by means of the expanded gradient (see Figure 4.10(b). However, the implicit surface initialization used in the fully guided approach (Figure 4.10(c)) further improves the convergence rate. Meanwhile, the unmodified method begins to approach the true answer only after 65 iterations (Figure 4.11). In a case where the true inclusion position is further from the starting salt boundary, I would likely have an even worse result using the unmodified approach.

CONCLUSIONS

I discuss the limitations of the standard level set formulation and how it constrains our ability to create salt models with inclusions. I then introduce the idea of using a likelihood map generated by interpreters as guidance for the inversion. This guidance is used to expand the level set gradient footprint, as well as initialize the implicit surface to create areas of higher sensitivity to updating. Using 2D synthetic examples, I show how the expanded gradient and the implicit surface initialization both contribute to speeding up the convergence.

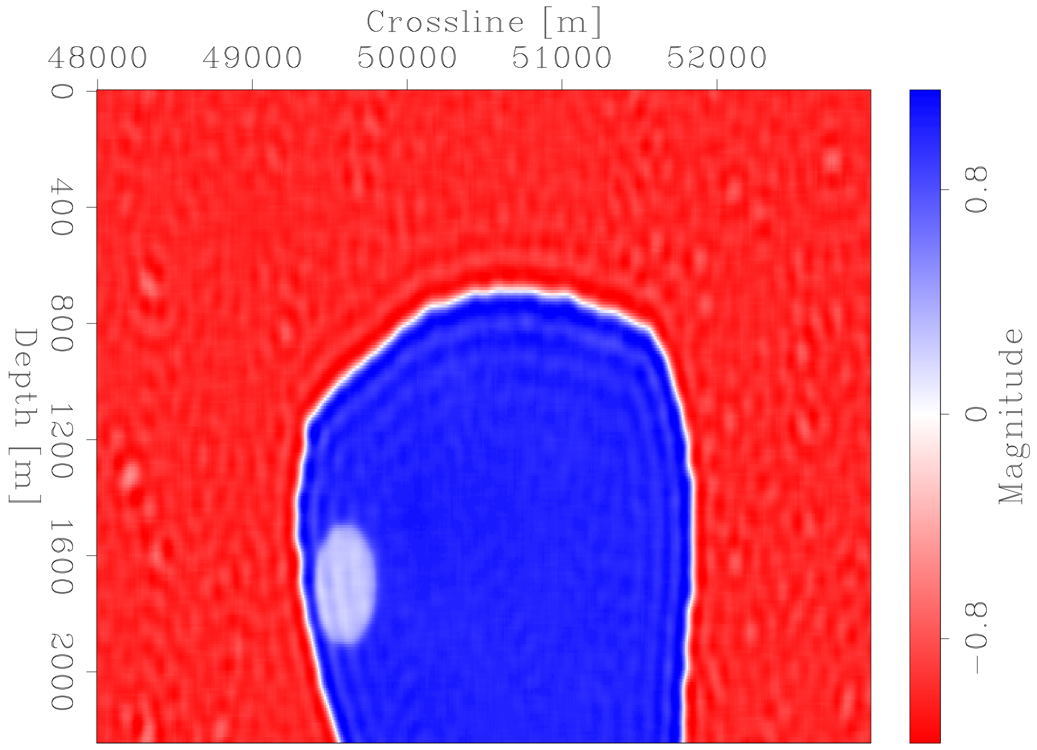


(a)

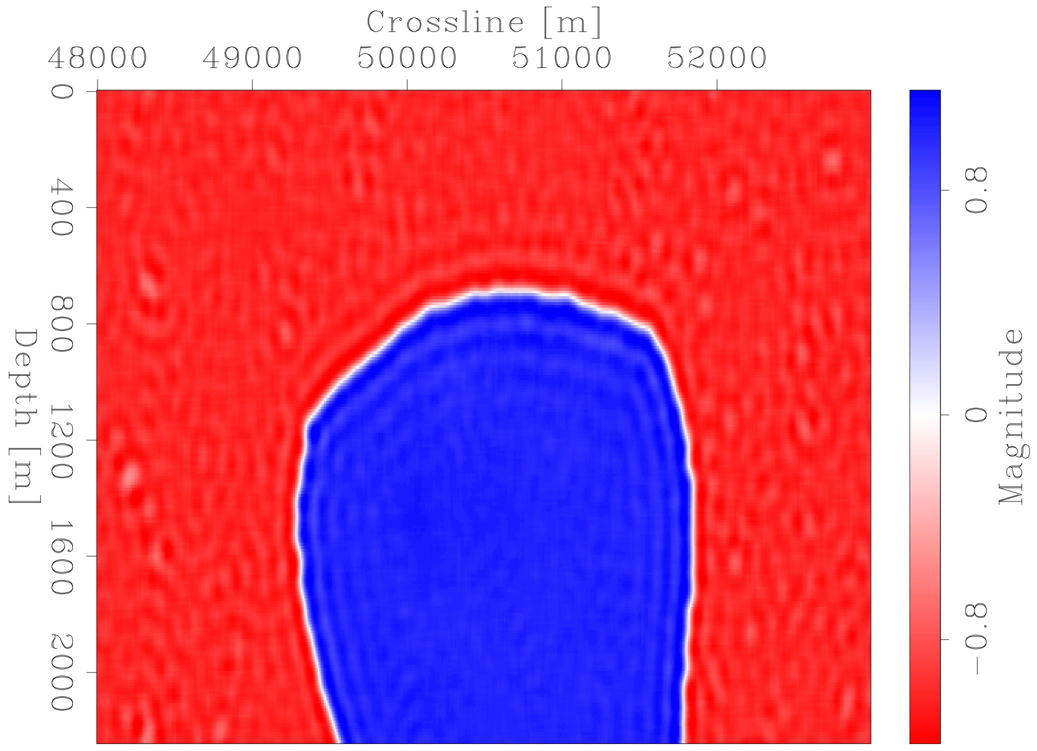


(b)

Figure 4.5: True model a) and starting model b) and used for all tests. [**ER**] chapter4/. truemodel-gom,startingmodel-gom

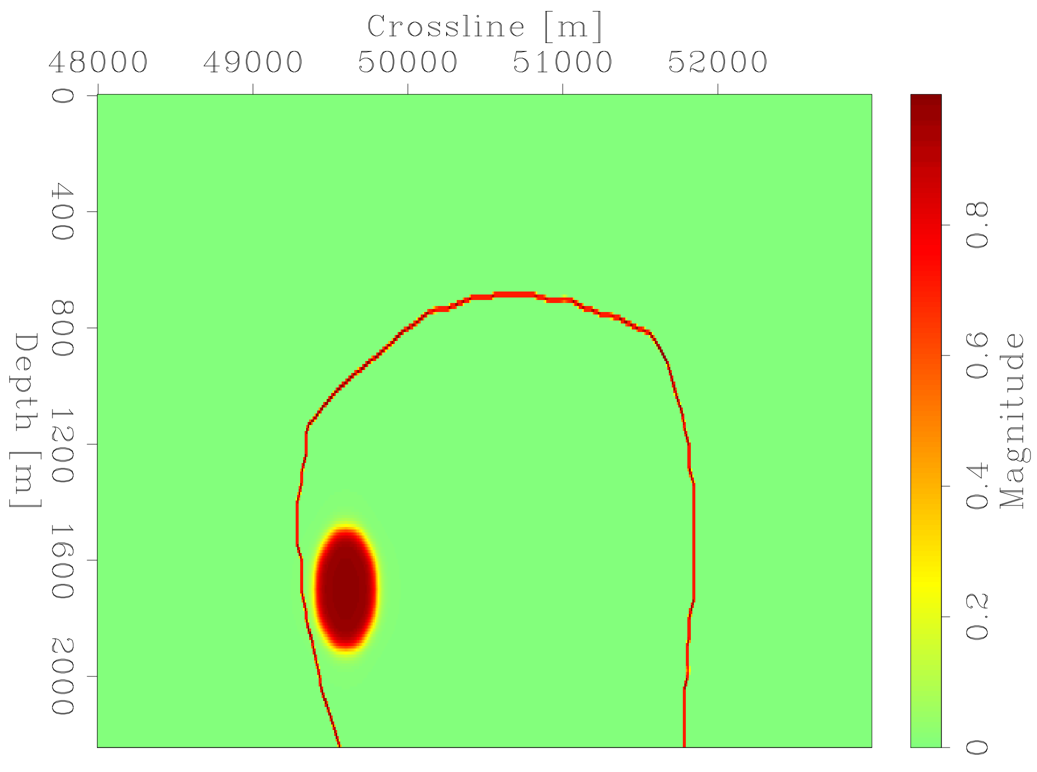


(a)

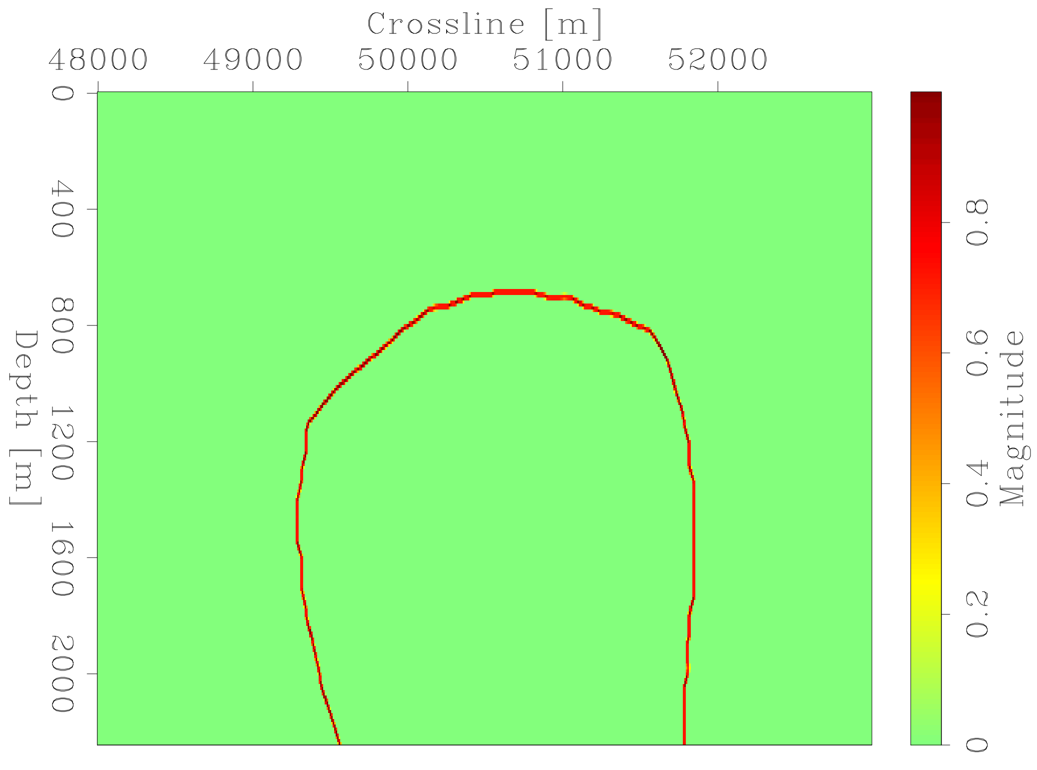


(b)

Figure 4.6: Implicit surface with interpreter guidance initialization a) and without it b). [**ER**] $\boxed{\text{chapter4/. initialphi-fullyGuided-gom,initialphi-unguided-gom}}$



(a)



(b)

Figure 4.7: Masking term ($\hat{\delta}(\phi_0, G)$) for guided gradient methods (a), and masking term ($\delta(\phi_0)$) for the unguided approach (b). [**ER**] chapter4/. guidedMask-gom,unguidedMask-gom

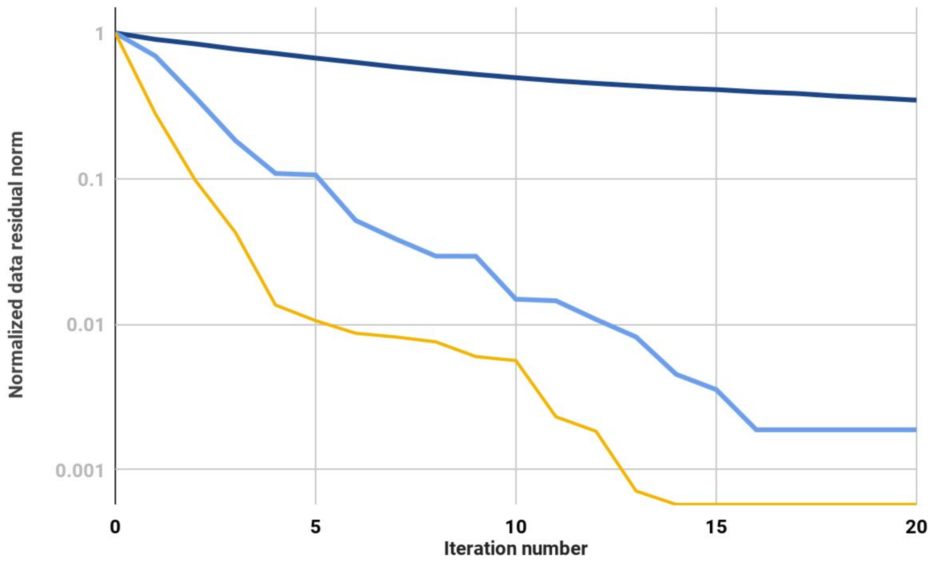


Figure 4.8: Data residual norm for level set inversion with unmodified approach (dark blue), partially guided approach (light blue), and fully guided approach (yellow). [**CR**] $\boxed{\text{chapter4/. gom-guidanceVsnoguidance-dataNorm}}$

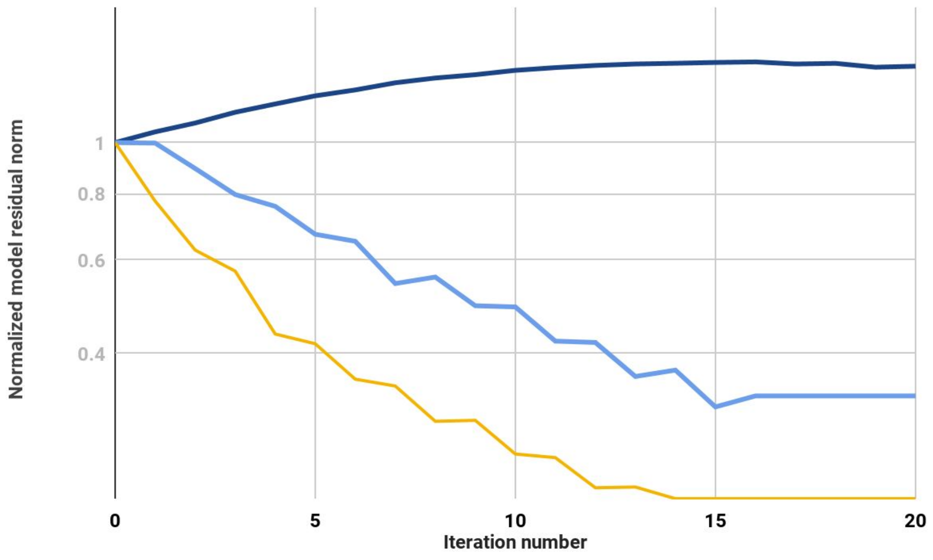
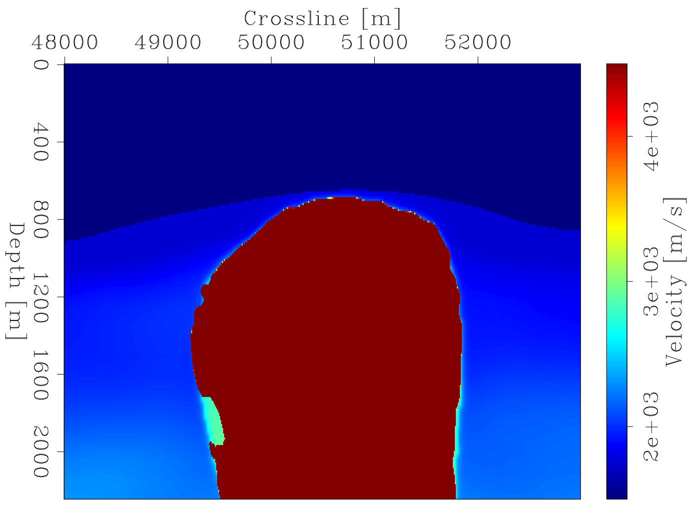
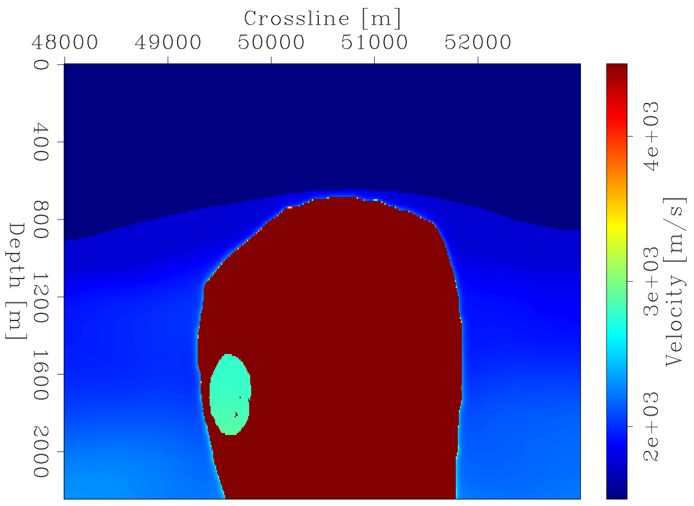


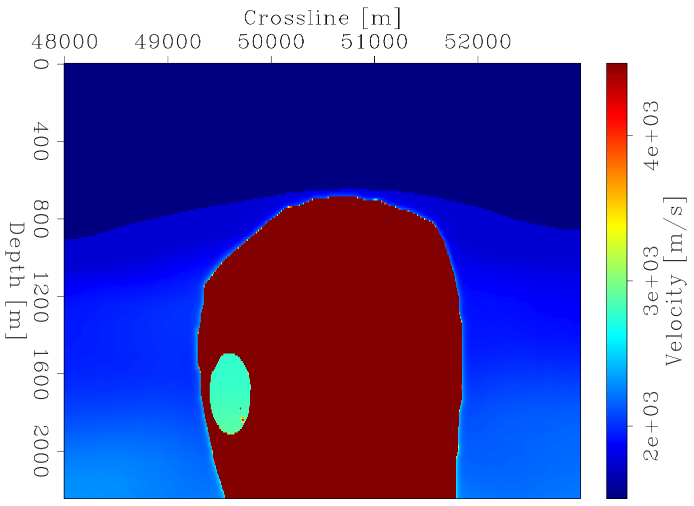
Figure 4.9: Model residual norm for level set inversion with unmodified approach (dark blue), partially guided approach (light blue), and fully guided approach (yellow). [**CR**] $\boxed{\text{chapter4/. gom-guidanceVsnoguidance-modelNorm}}$



(a)



(b)



(c)

Figure 4.10: Inverted model after 15 steepest descent iterations using unguided approach a), partially guided approach b), and fully guided approach c). **[CR]**
chapter4/. model-15-unguided,model-15-partiallyGuided,model-15-fullyGuided

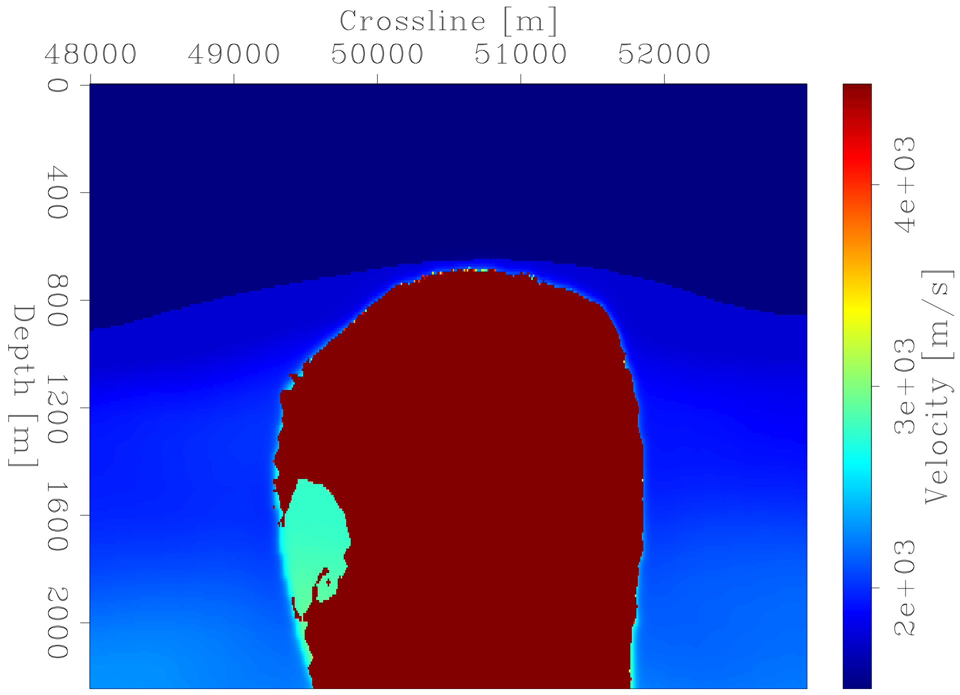


Figure 4.11: Inverted model after 65 steepest descent iterations using unmodified approach. [**CR**] $\boxed{\text{chapter4/. model-65-unguided}}$

Chapter 5

Application to 3D Gulf of Mexico field data

In order to demonstrate the theory and algorithms introduced in previous chapters on a convincing example, I apply level set inversion to a 3D Gulf of Mexico ocean bottom node (OBN) data set provided by Shell. I begin this chapter by describing the dataset and its context, and explore the RTM imagery provided to me for clues related to the actual earth model. Next, I demonstrate how I identify a subset of nodes to use for my inversion (based on my own RTM images) to improve computation turnaround. After, I discuss the necessity of matching amplitudes with the observed data, and discuss how I modify the objective function by weighting the residuals and incorporate a kinematic virtual source operator into the gradient calculation. Following, I discuss the inversion workflow used and the resulting inverted model. To conclude, I show the improvements in the migrated image that was created using the new (inverted) velocity model.

DATA SET OVERVIEW

The 3D dataset that I use was acquired in the Gulf of Mexico and provided by Shell Exploration Inc. using Fairfield Z3000 ocean-bottom nodes (Figure 5.1). The data was collected in 2010 to improve imaging around the salt by leveraging wide-azimuth acquisition with ocean bottom node technology. I was provided with the pressure, horizontal, and vertical components of the data. The field sits in the Garden

Banks region about 362km south-west of New Orleans, Louisiana (Figure 5.2) in approximately 830 m of water, with an airgun shooting footprint covering an area of about 48x48km. The reservoir itself sits beneath thick layers of salt more than 6 km below the sea floor, and production began on the field in 2014. In the region near the salt diapir, the node footprint coverage is just enough to cover the salt (see Figure 5.5). The area of production is centered around a salt protrusion that nearly reaches the water bottom (Figure 5.3).

Figure 5.1: Fairfield Z3000 ocean bottom node used to record data. [**NR**] chapter5/. Z3000-node

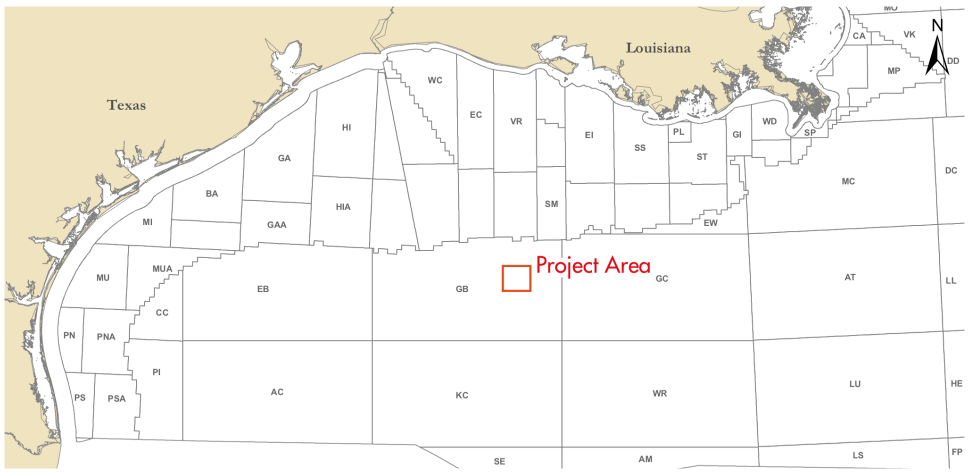


Figure 5.2: Map showing approximate project location in the Garden Banks region of the Gulf of Mexico. [**NR**] chapter5/. overview-map

Along with the data, Shell provided a Reverse Time Migration (RTM) image cube produced from the downgoing pressure data. The approach they used is called mirror imaging, since all downgoing data recorded on the OBN units (besides the direct arrival) is a result of the mirror-like reflection of subsurface events off the ocean surface (Grion et al., 2007). When looking at this mirror-image, one can see an inclusion in the salt at about 1,700m deep, with an inline position of 214,800m

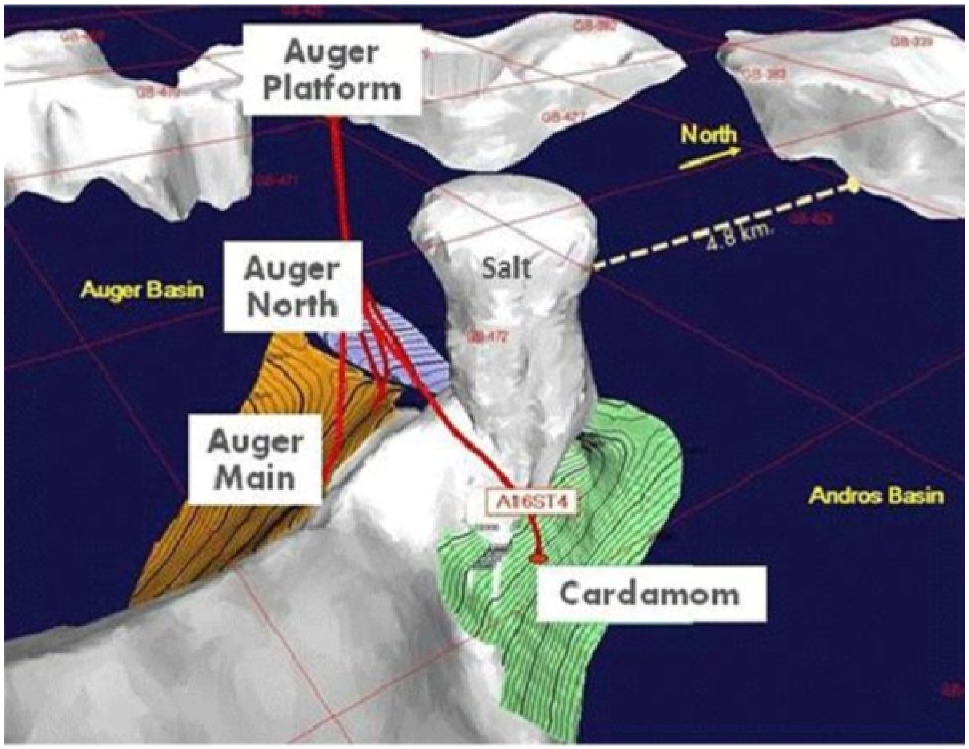


Figure 5.3: Oblique view of the salt body that is most prominent in the dataset I investigate. [**NR**] chapter5/. cardamom-salt-diagram

and crossline position of 49,600m (see Figures 5.6, 5.7, 5.8). When comparing against the velocity model provided (Figures 5.9(a), 5.9(b)), one can see that the inclusion observed was not included in the velocity model. I intend to show that my level set FWI inversion workflow can help recover a velocity model that takes this inclusion into account, and correspondingly produces a more accurate RTM image.

OPTIMIZING INCLUSION ILLUMINATION

Because of limited computational resources, I am interested in minimizing the number of wavefield propagations needed to perform inversion. By taking advantage of reciprocity between the nodes and sources for the hydrophone component, one can change the modeling to use node positions for the sources, and shot positions at the ocean surface as recording locations (Knopoff and Gangi, 1959). While this does assume the signature of shots in the field experiment are consistent with each other, I find that this assumption is valid, and exploit the advantages of reciprocity in this

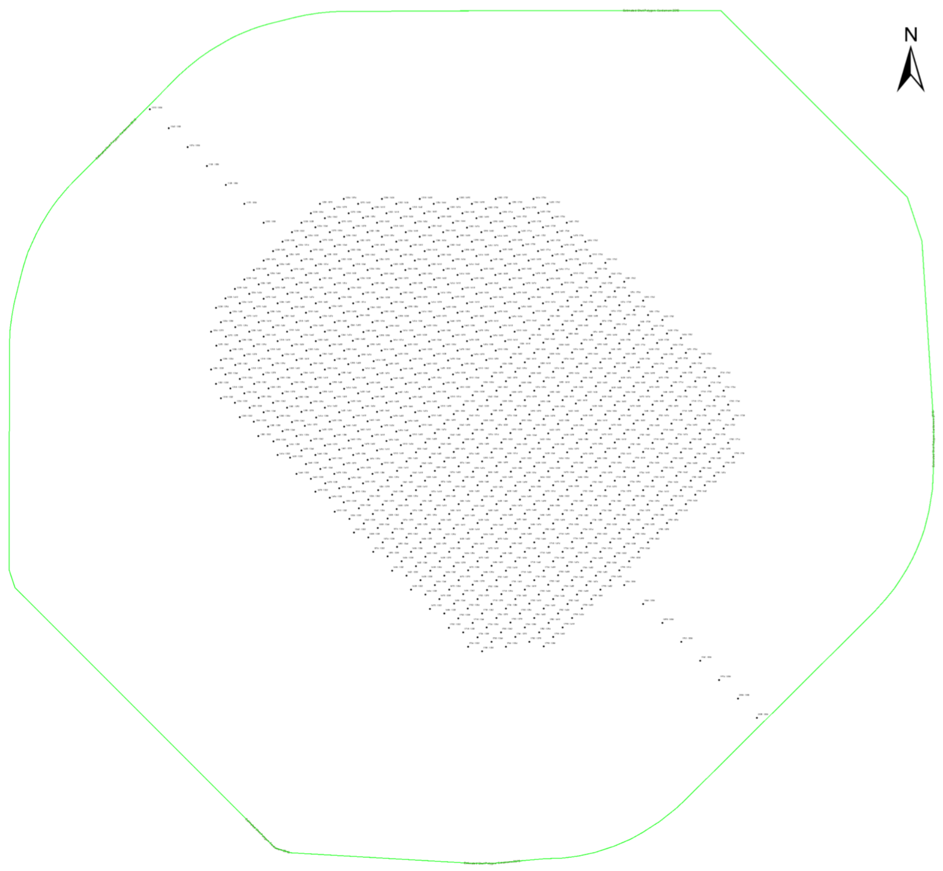


Figure 5.4: Map showing the boundary of the extent of the airgun shooting (green line) and the OBN positions (black points). [**NR**] chapter5/. full-acquisition-map

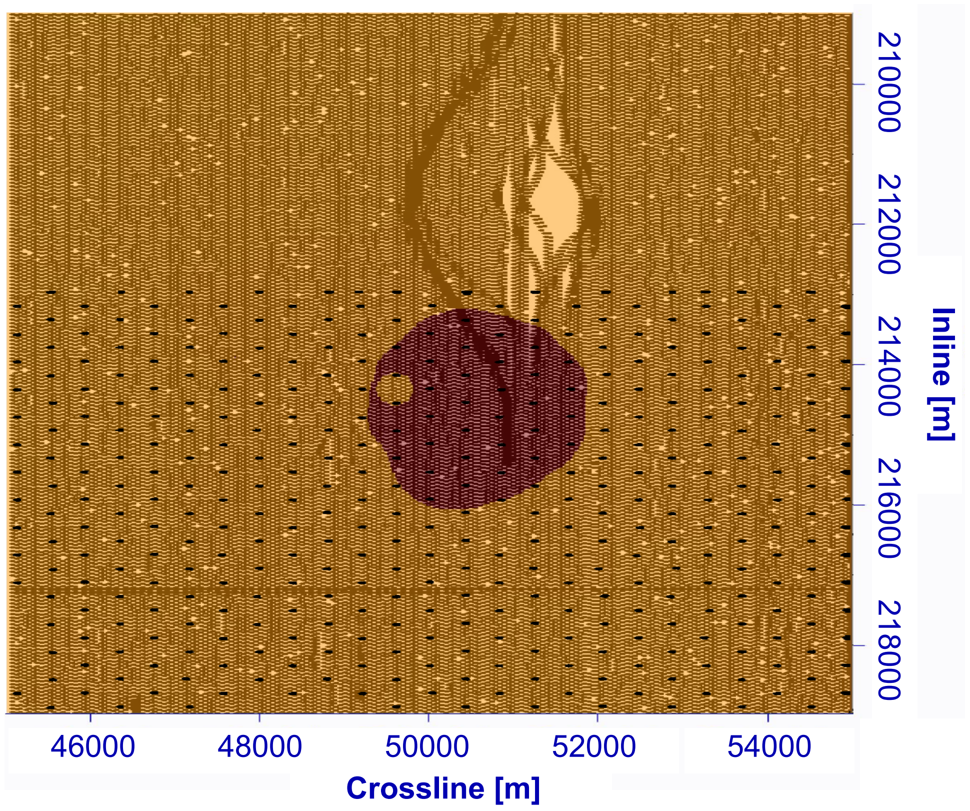


Figure 5.5: Map showing acquisition geometry in the greater region surrounding the zone where inversion and imaging were performed (depth slice at 1,800m). Red is the salt body with approximate inclusion location, black dots are node positions, and grey dots are shot positions. One easily can see where the boats divert around a production platform at 211,700m inline position. [**NR**] chapter5/. salt-acquisition-mapT

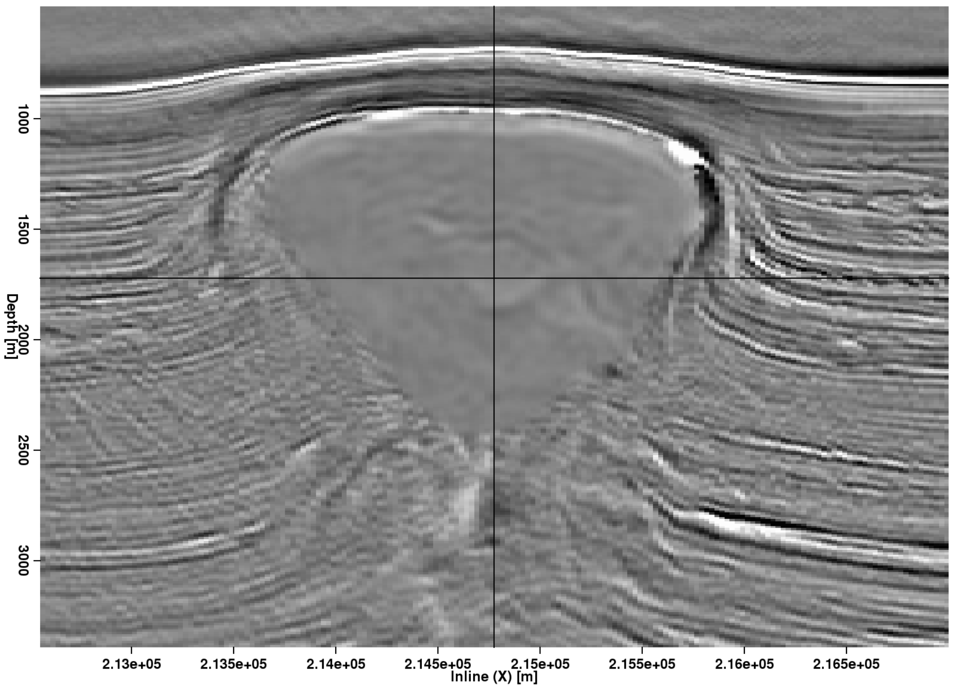


Figure 5.6: Original mirror image RTM provided by Shell; front view. [**NR**]
chapter5/. original-shell-rtm-FRONT

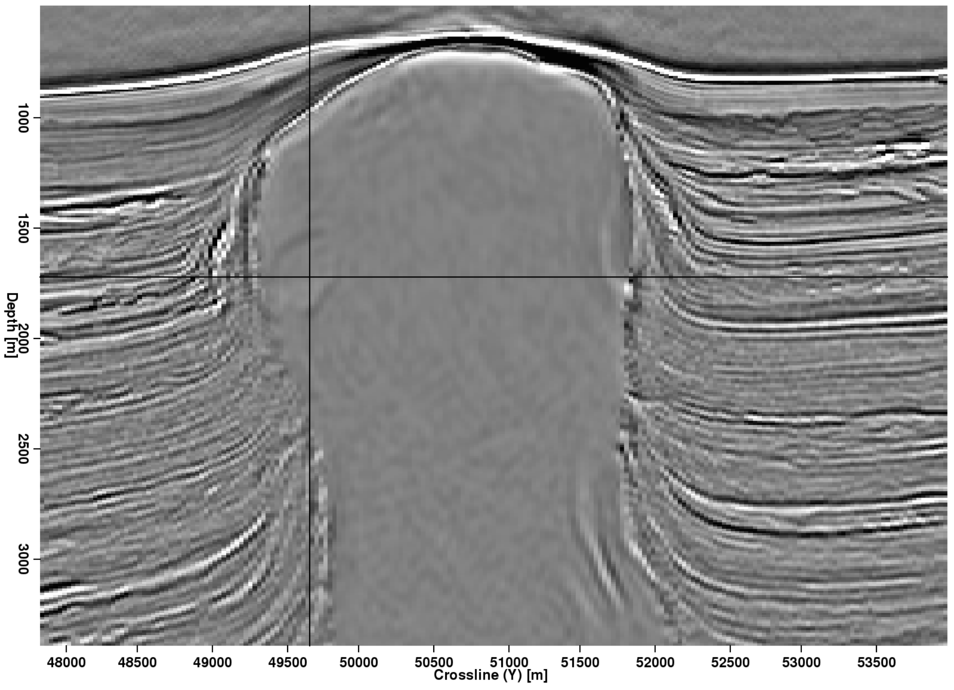


Figure 5.7: Original mirror image RTM provided by Shell; side view. [**NR**]
chapter5/. original-shell-rtm-SIDE

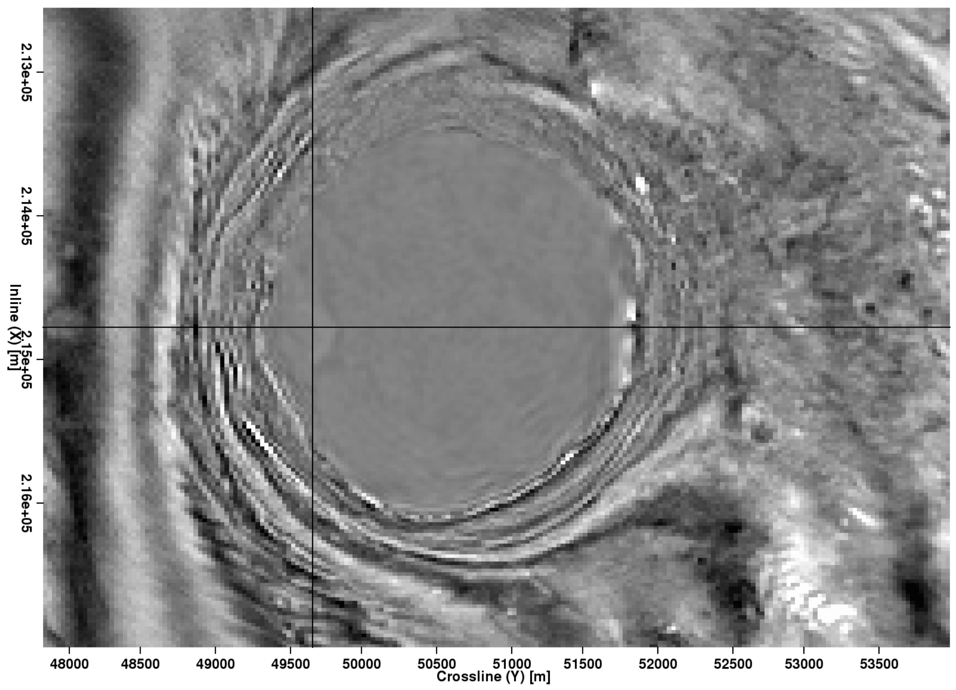


Figure 5.8: Original mirror image RTM provided by Shell; top view. [**NR**]
chapter5/. original-shell-rtm-TOP

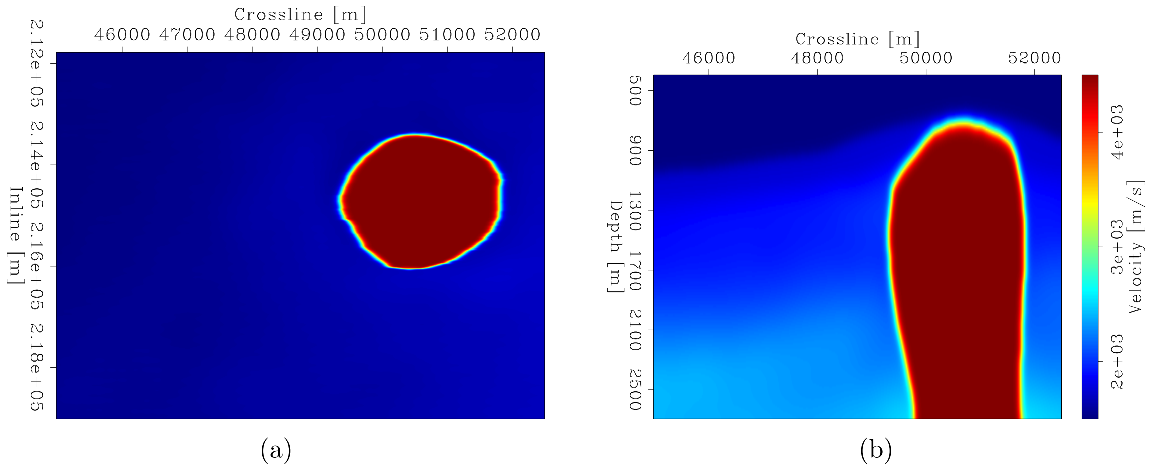


Figure 5.9: Velocity model used for migration with 10m grid spacing. Figure (b) at 1,700m deep; Figure (a) at inline position 214,800m.
chapter5/. migvel-top,migvel-side

dataset.

While I limit the propagation domain to a ∼7.5x7.5km region centered around the inclusion, I ignore nodes that do not readily contribute to illumination in the inclusion area of interest (AOI) (Figures 5.10 and 5.11). I do this by performing RTM with all the nodes and shots in my model domain, and use the downgoing component of the hydrophone data. Next, I select with a masking function the AOI (in our case, immediately around the salt inclusion). After, I cross-correlate the RTM image produced by each node-gather with the area of interest from the full-stack RTM image. This gives me information about which nodes correlate most strongly with the full-stack image. I then sort the RTM node-gather images by this correlation statistic, and select those that have the highest correlation while ignoring those that do not.

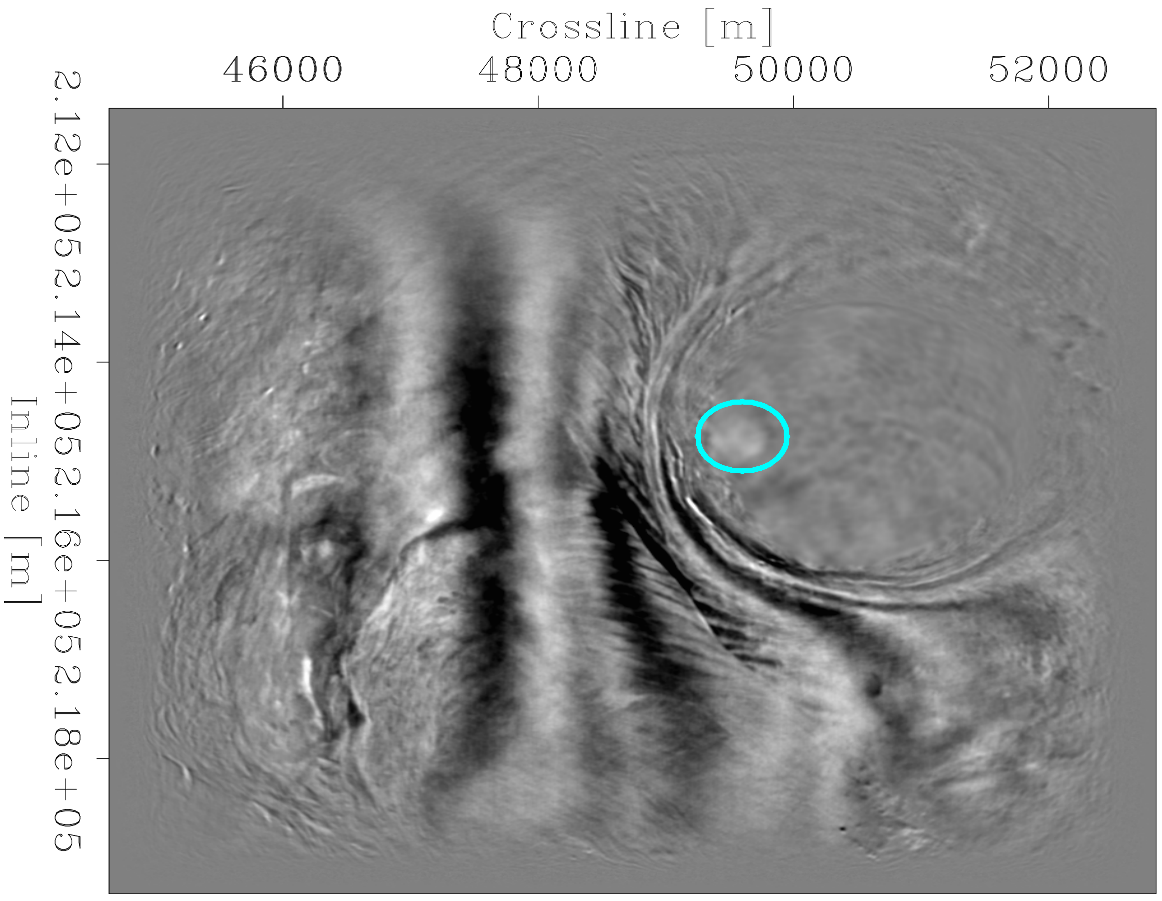


Figure 5.10: Area of interest (cyan) used in cross-correlation overlaid on full-stack mirror RTM image. Slice at depth 1,770m. chapter5/. ideal-aoi-top

By comparing the RTM images using higher numbers of nodes (Figures 5.12,

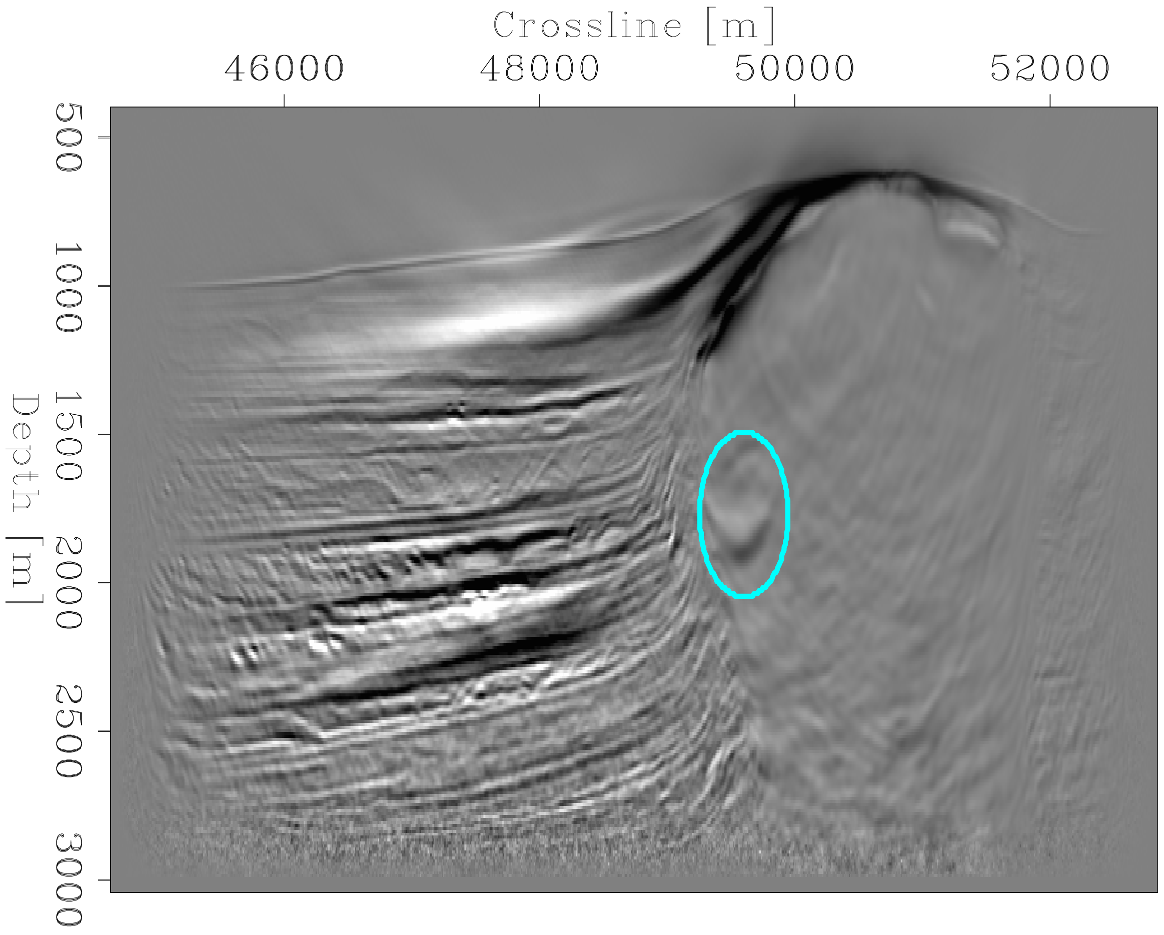


Figure 5.11: Area of interest (cyan) used in cross-correlation overlaid on full-stack mirror RTM image. Slice at inline position 214,800m. chapter5/. ideal-aoi-side

5.13, 5.16, 5.17) with those using fewer nodes (Figures 5.14, 5.18, 5.15, 5.19), it is easy to see how image quality decreases as nodes are removed. The goal is to choose the lowest number of nodes that still correspond to a reasonably good RTM image. I ultimately chose to use the node subset that comprises the top 35 percent of (positively correlated) illumination contribution to the region around the inclusion, which amounts to 78 of the 288 original nodes being used.

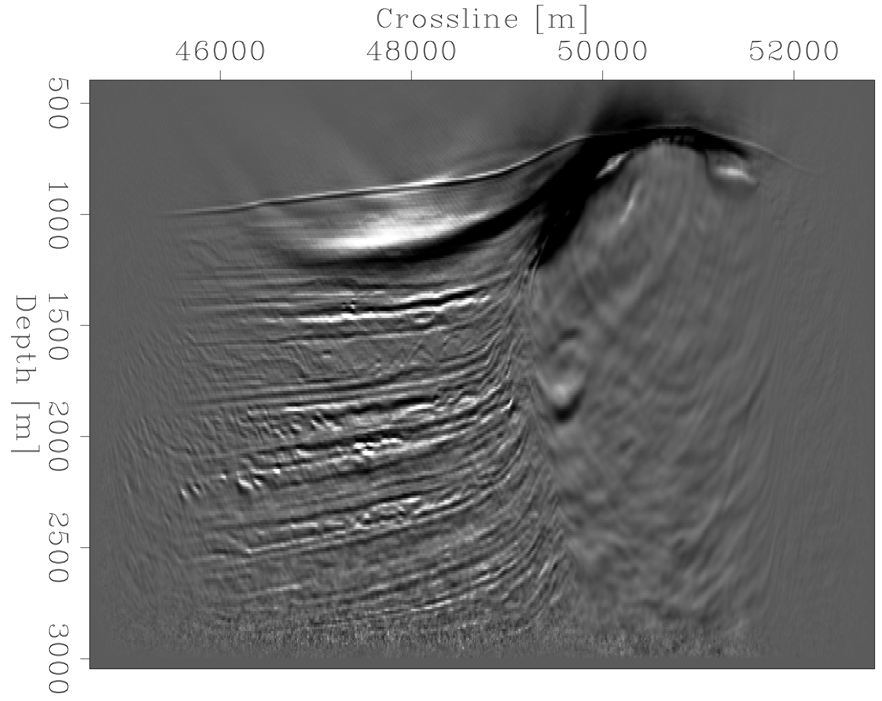


Figure 5.12: RTM image from stacking top 50 percent of positively correlated node-gather images (214,800m inline position). chapter5/. selective-RTMdown-side-stack-50

PHASE ONLY OBJECTIVE FUNCTION AND ITS IMPLICATIONS

Convergence using the level set objective function as I have formulated it will be partly determined by the ability of the modeling operator to match the phase and amplitudes in the observed data. The Green's function of the earth produces an elastic response, but the modeling operator I use assumes an acoustic response. I chose to model acoustically to avoid the added computational expense of modeling elastic waves. This means that my operator will have difficulty matching the amplitudes

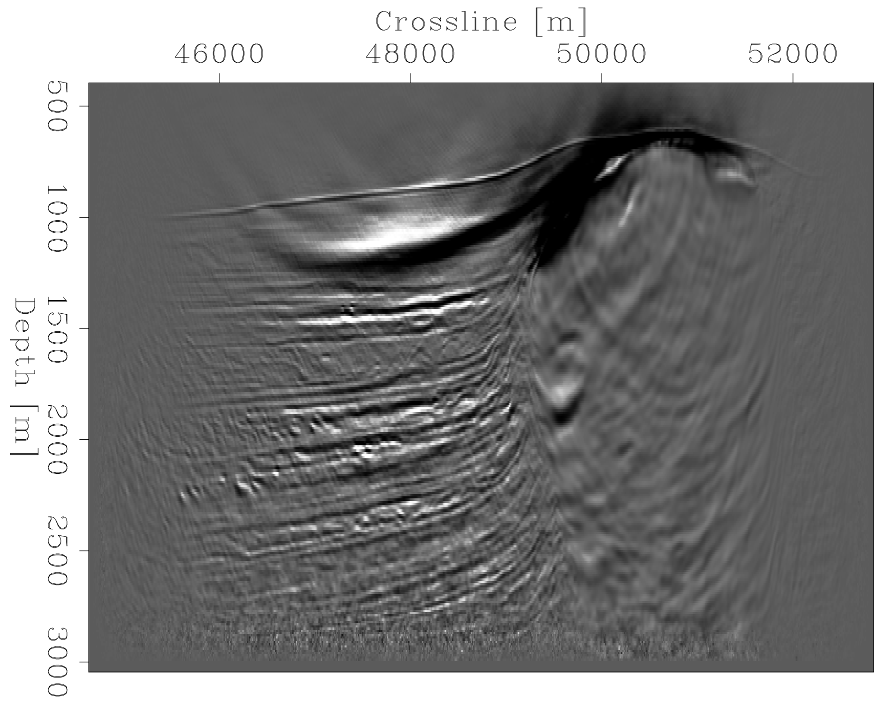


Figure 5.13: RTM image from stacking top 35 percent of positively correlated node-gather images (214,800m inline position). chapter5/. selective-RTMdown-side-stack-35

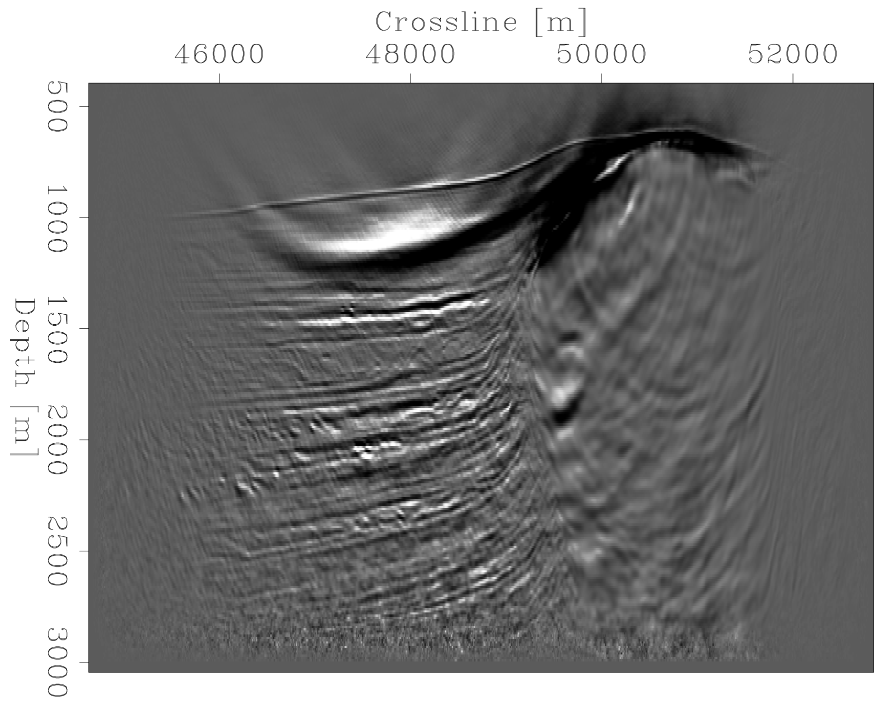


Figure 5.14: RTM image from stacking top 25 percent of positively correlated node-gather images (214,800m inline position). chapter5/. selective-RTMdown-side-stack-25

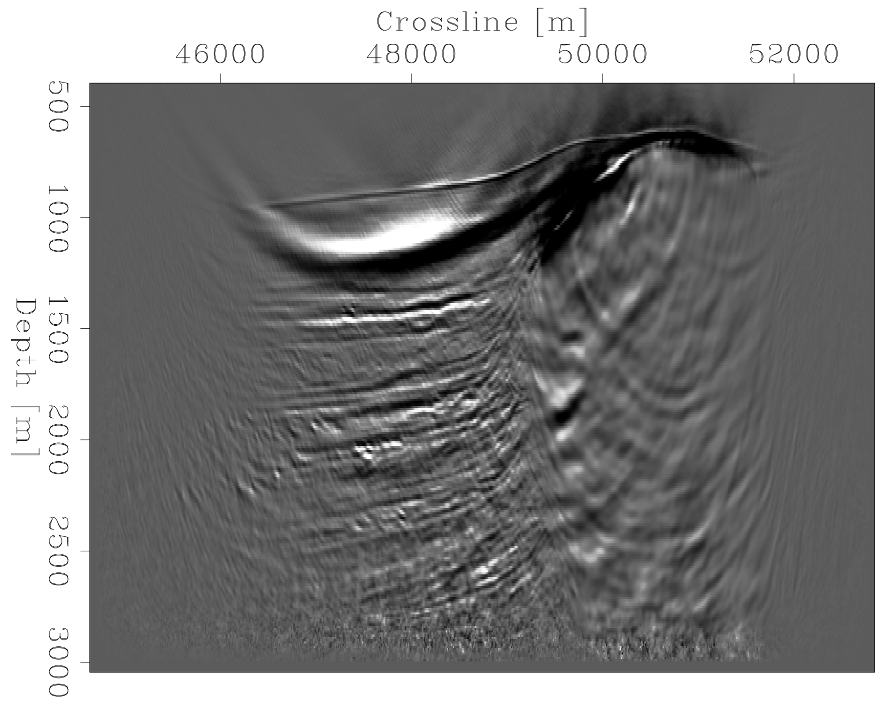


Figure 5.15: RTM image from stacking top 15 percent of positively correlated node-gather images (214,800m inline position). chapter5/. selective-RTMdown-side-stack-15

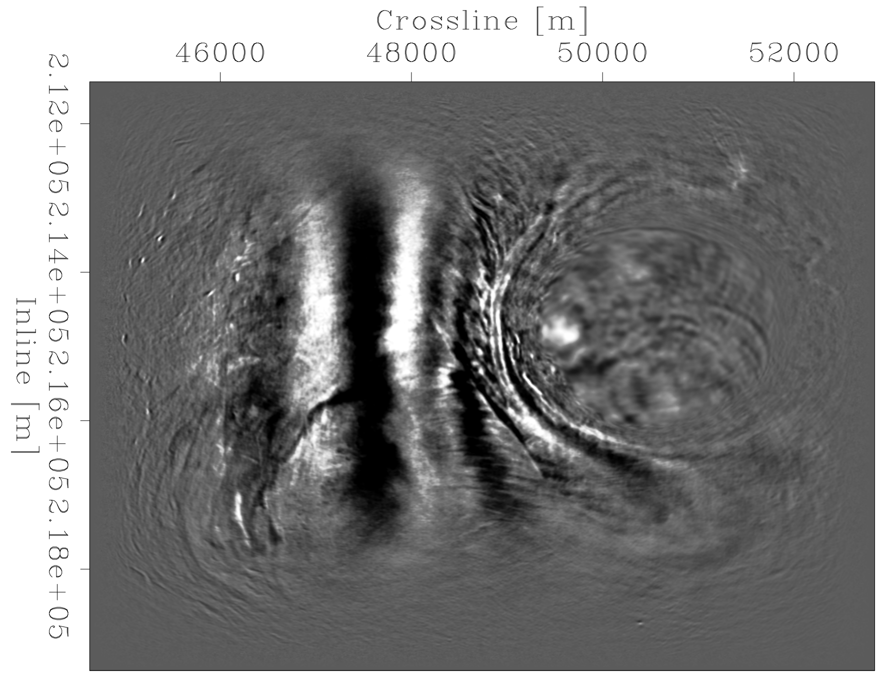


Figure 5.16: RTM image from stacking top 50 percent of positively correlated node-gather images (1,770m inline position). chapter5/. selective-RTMdown-top-stack-50

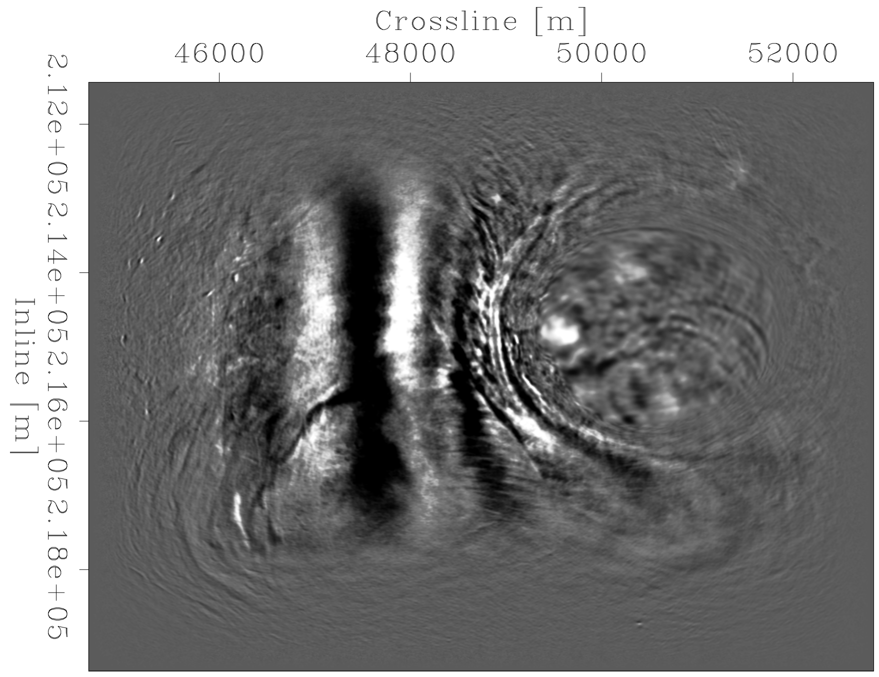


Figure 5.17: RTM image from stacking top 35 percent of positively correlated node-gather images (1,770m inline position). chapter5/. selective-RTMdown-top-stack-35

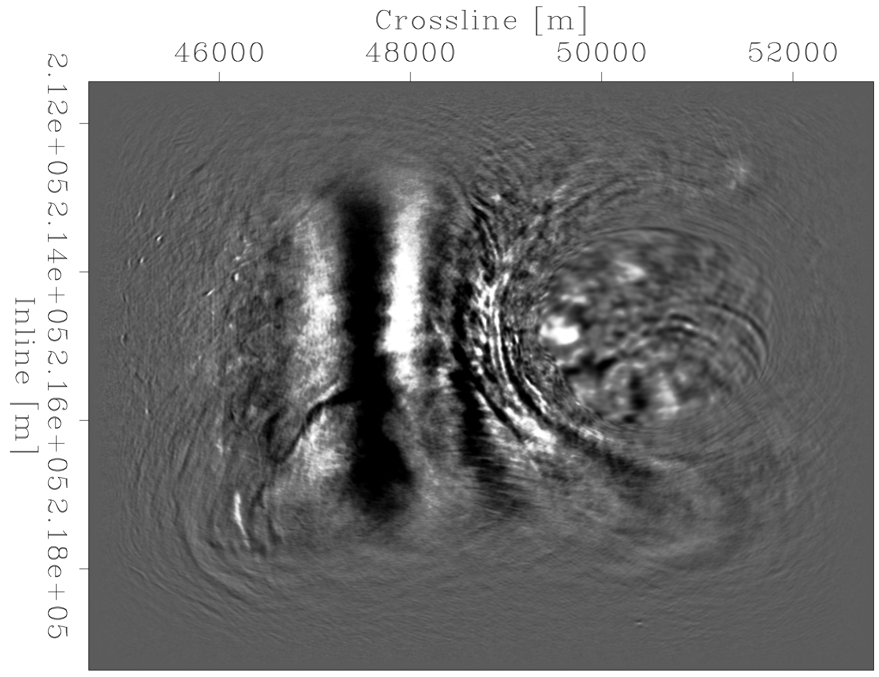


Figure 5.18: RTM image from stacking top 25 percent of positively correlated node-gather images (1,770m inline position). chapter5/. selective-RTMdown-top-stack-25

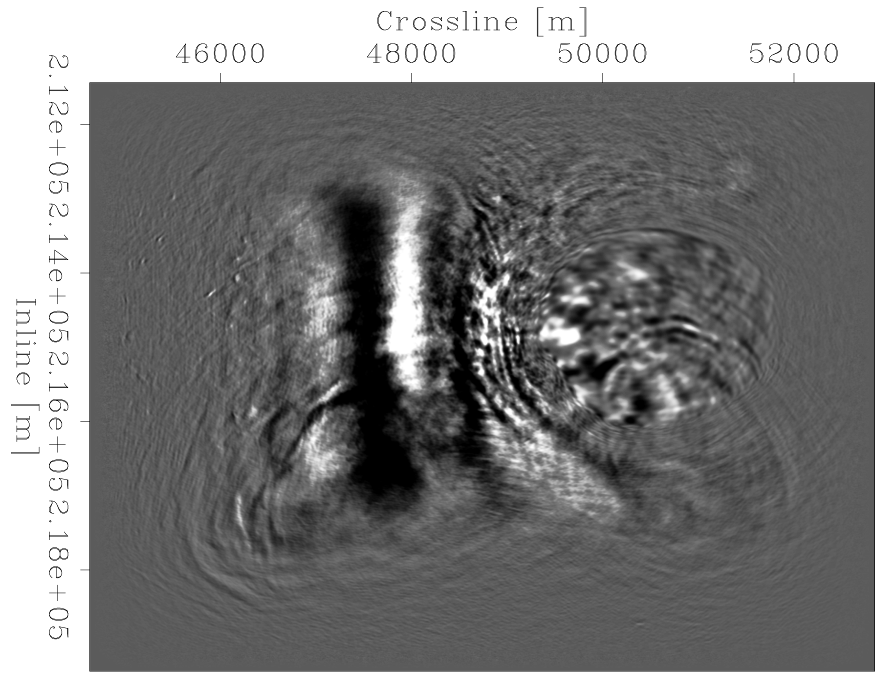


Figure 5.19: RTM image from stacking top 15 percent of positively correlated node-gather images (1,770m inline position). chapter5/. selective-RTMdown-top-stack-15

found in the observed data, even if I have the exact earth model.

For this reason, I modify the objective function to make it agnostic to amplitude information and become primarily based on minimizing kinematic errors. I follow the work done in Shen (2010) and adopt an objective function (ψ) that includes trace-by-trace normalization of the data:

$$\psi(m) = \frac{1}{2}\sum_{s,g} \|r_d(s,g)\|^2 \tag{5.1}$$

$$\psi(m) = \frac{1}{2}\sum_{s,g} \left\| \frac{F(m,s,g)}{\sqrt{F(m,s,g)^T F(m,s,g)}} - \frac{d_{obs}(s,g)}{d_{obs}(s,g)^T d_{obs}(s,g)} \right\|^2, \tag{5.2}$$

where r_d is the normalized residual, m is the velocity model, d_{obs} is the field data, $F(m,s,g)$ is the acoustic modeling operator, and s and r are the shot and receiver indices respectively. I take the derivative of this new objective function to find a new gradient calculation:

$$J_f(m) = r_d^T(s,g)\frac{\partial\left(\frac{F(m,s,g)}{\sqrt{F(m,s,g)^T F(m,s,g)}}\right)}{\partial m} \tag{5.3}$$

$$J_f(m) = r_d^T(s,g)\frac{\partial\left(\frac{F(m,s,g)}{\sqrt{F(m,s,g)^T F(m,s,g)}}\right)}{\partial F(m,s,g)}\frac{\partial F(m,s,g)}{\partial m}. \tag{5.4}$$

I represent this as a series of operators:

$$J_f^T(m) = B^T P^T r_d(s,g), \tag{5.5}$$

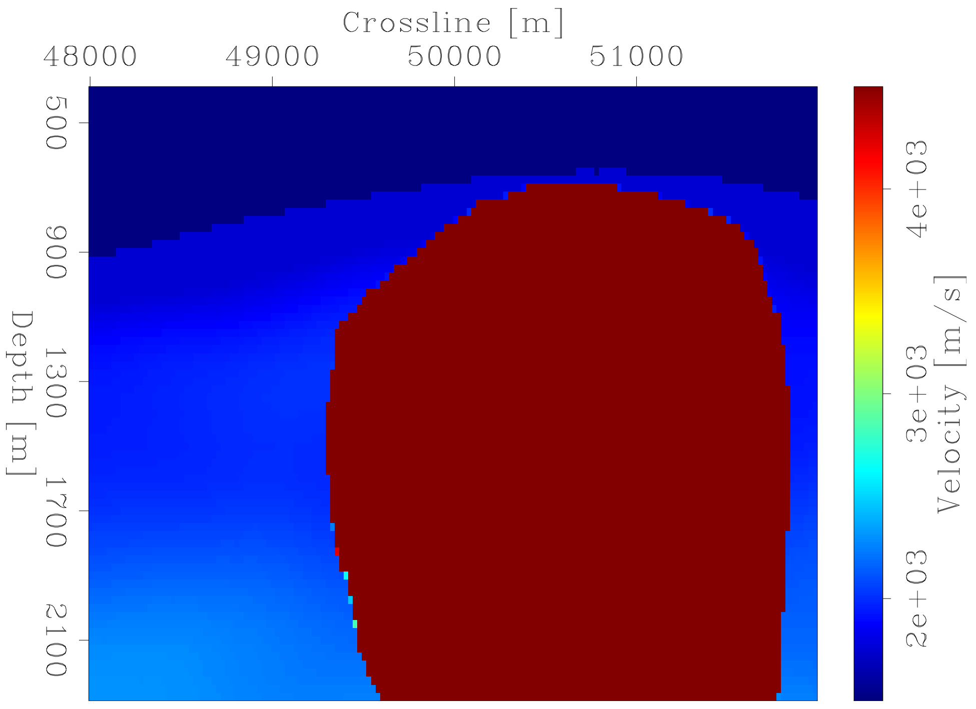
where B is the born operator, and P is a new operator to account for the modifications made to the objective function. For this reason, the Gauss-Newton Hessian computation is also altered to account for P:

$$H_{GN} = PBB^T P^T. \tag{5.6}$$

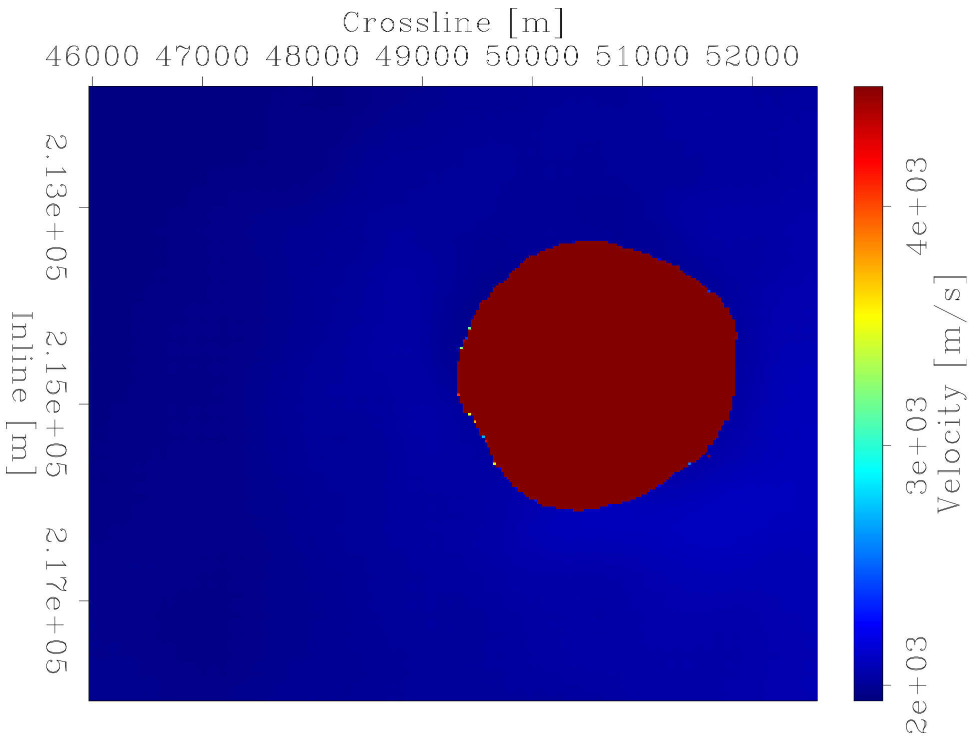
INVERSION IMPLEMENTATION

Using the nodes selected as described in the previous section, I ran an inversion using a starting velocity model very similar to the smooth one used in migration (Figures 5.20(a), 5.20(b)). This model extends into negative depth (above the water surface) since mirror wave propagation was used in the inversion. The algorithm used alternates between updating the salt boundary (level set) and the background velocity for each non-linear (outer-loop) iteration. Within each outer-loop iteration is an iterative inversion to find the search direction in either the level set or background velocity space using the Gauss-Newton Hessian. After this, a line search finds the optimal scaling parameter to apply to the search direction and then update the model. The full workflow used is described in Algorithm 2. For the data itself, I performed designature of the hydrophone and vertical components, PZ-summation to create the downgoing separated data, and a shaping filter to remove the bubble. These pre-processing steps are described in detail in Appendix A and B.

Since the RTM images I created (Figures 5.10, 5.11) and those from Shell (Figures



(a)



(b)

Figure 5.20: Inversion starting velocity model side view (a) and top view (b). chapter5/. initvel-side,initvel-top

5.6, 5.7,5.8) agree on the presence of some kind of inclusion in the salt, I begin the inversion by preferentially initializing the implicit surface in the region where I believe it to be (Figures 5.21, 5.23). For the first non-linear iteration, I also use guidance to extend the gradient footprint as described in chapter 4 (see Figure 5.24(a)). Because of this guidance, the first inversion iteration is capable of pushing the implicit surface value below zero to create a level set marking the inclusion. However, for following iterations, I do not use interpreter guidance to extend the gradient. In the case where an inclusion takes shape because of the first update, then the standard level set gradient is active around the inclusion edge and is sufficient to adjust it. In following iterations, the gradient can close or open the inclusion further without the need for an extended gradient (see Figure 5.24(b)). It is only the first iteration where that extension is necessary. In the case where the first update does not begin an inclusion, I assume that the data does not support the existence of one after all.

Because the nodes chosen for inversion are focused on illuminating the inclusion area, I limit the extent of the level set updating to the inclusion area of interest (see Figure 5.22). By doing this, I reduce spurious updating of the salt boundary in regions that receive poor illumination by the chosen acquisition geometry, which would likely be driven by artifacts in the gradient.

After running this alternating inversion algorithm for 35 iterations, I find that an inclusion was created in the model, and that the background velocity model (which includes the inclusion velocity) has been updated in a manner that is more geologically consistent (Figures 5.25, 5.26). Taking the difference between the beginning and final models gives a better view of the updates made, highlighting a shift in the ocean bottom interface, and adding some higher velocity zones near the top of salt (Figures 5.27, 5.28). By looking at the difference between the starting and final background velocity model, we can see that the inclusion velocity has decreased about 130[m/s] (from a starting guess velocity of 4250[m/s]) (Figures 5.29, 5.30). The objective function value shows a steady decrease over non-linear iterations, with the predominant decreases occurring from updates in the background velocity, which have a more significant impact on events in the data space (compare Figure 5.31 and 5.32).

In order to validate an actual improvement in the velocity model, I perform RTM on both the initial model (unsmoothed) and the final model after inversion and compare the two images. The area where one would expect to see the inclusion make the

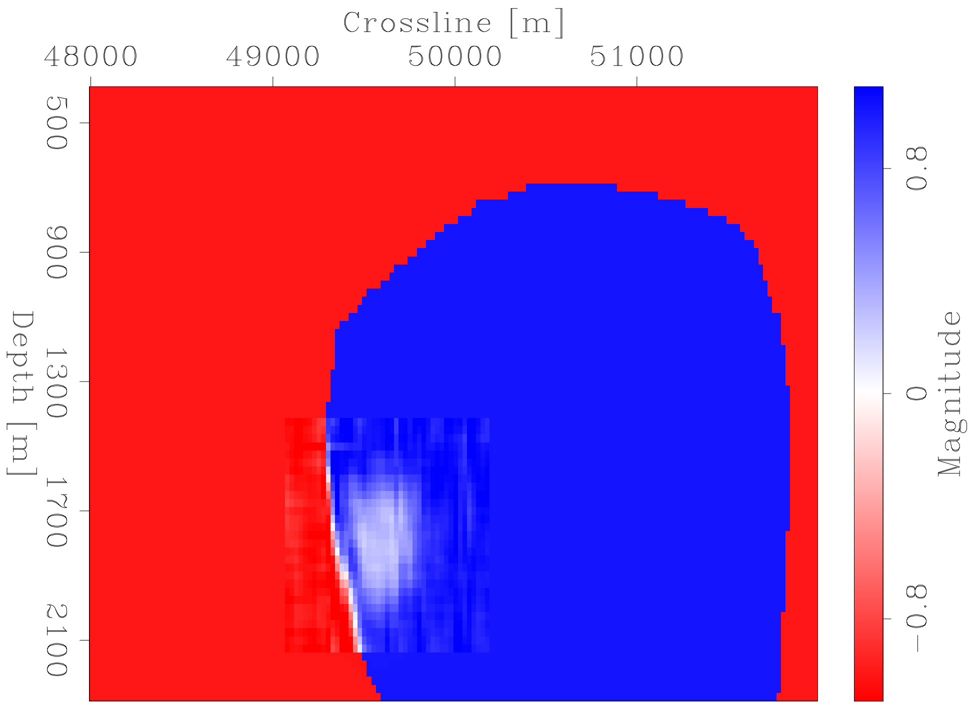


Figure 5.21: Initial implicit surface (214,800m inline position). chapter5/. phi-side-it-0

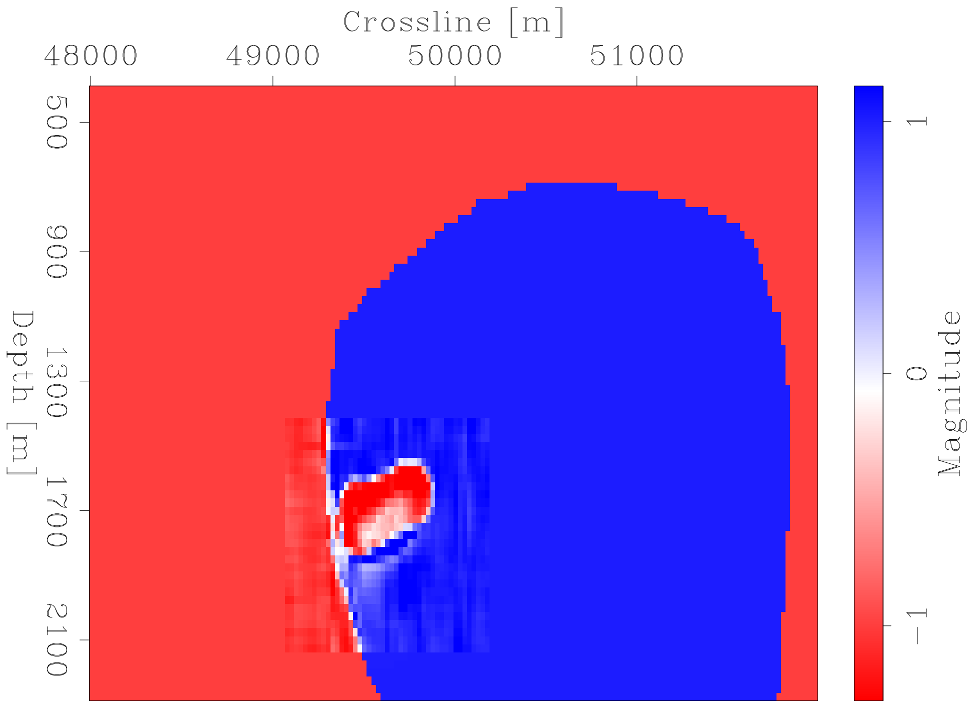


Figure 5.22: Implicit surface after 35 iterations of inversion (214,800m inline position). chapter5/. phi-side-it-35

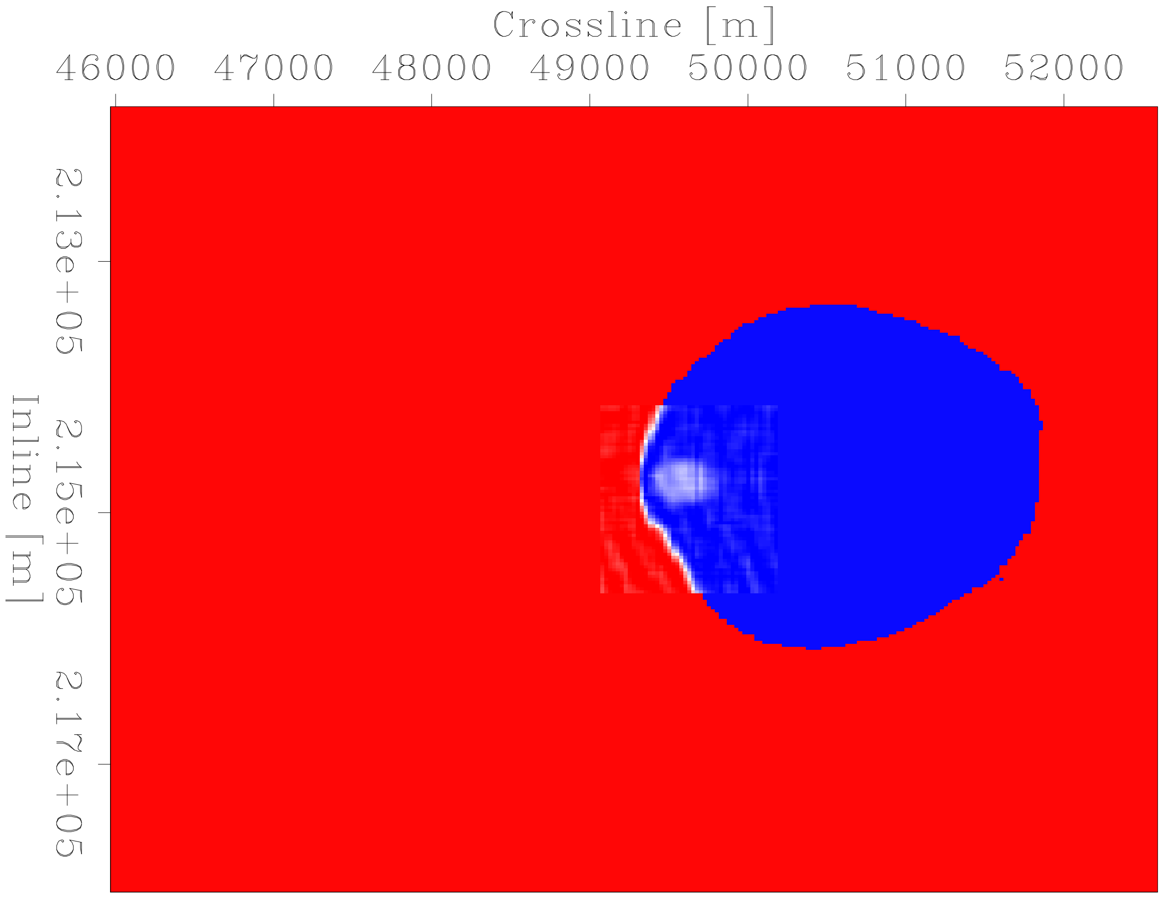
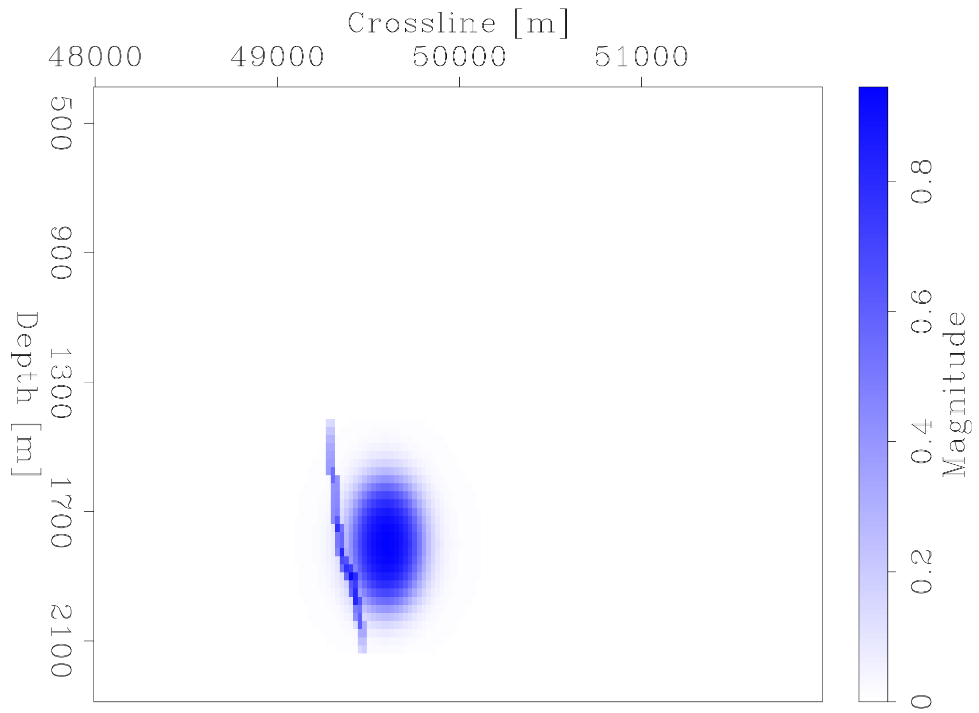
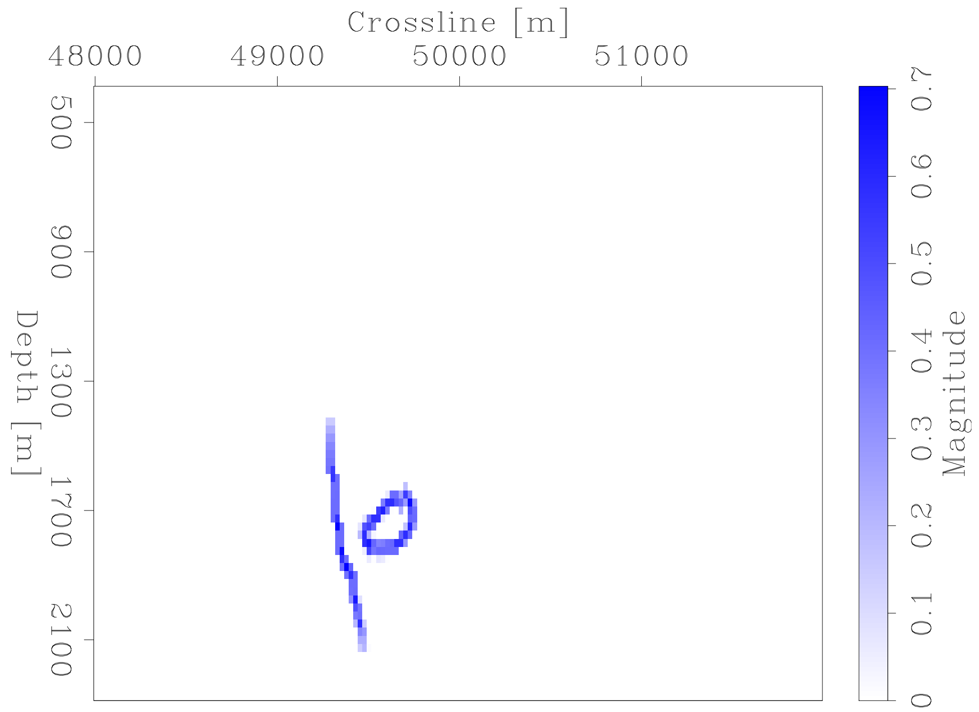


Figure 5.23: Initial implicit surface (1,700m depth position). chapter5/. phi-top-it-0



(a)



(b)

Figure 5.24: Masking used for first level set iteration (a) and for second level set iteration (b). Masking in (a) is based off of $\delta(\phi)$, while masking in (b) is based off of $\delta(\phi, G)$. chapter5/. saltmask-side-it-0,saltmask-side-it-2

Algorithm 2 Alternating Gauss-Newton Hessian updating algorithm

```
1: procedure LevelSetInversion-order2( $d_{\text{obs}}$,$\phi_0$,$b_0$ )
2:     for $i$ in (1,$N$) do
3:         $d_{\text{syn}}(i) = \text{F}(\phi_i, b_{i-1})$
4:         $\triangle d_i = d_{\text{obs}} - d_{\text{syn}}(i)$
5:         $g_i = D^T B^T \triangle d_i$
6:         if EvenNumberedIteration then
7:             $\triangle \lambda_i = \textbf{CGGNHessianInvSalt}(g_i)$
8:             $\triangle \phi_i = \text{D}(\triangle \lambda_i)$
9:             $\triangle b_i = 0$
10:            $\alpha = \textbf{linesearch}(\triangle \phi_i)$
11:            $\beta = 0$
12:        else
13:            $\triangle \phi_i = 0$
14:            $\triangle b_i = \textbf{CGGNHessianInvBack}(g_i)$
15:            $\alpha = 0$
16:            $\beta = \textbf{linesearch}(\triangle b_i)$
17:        end if
18:        $\phi_i = \phi_{i-1} - \alpha \cdot \triangle \phi_i$
19:        $b_i = b_{i-1} - \beta \cdot \triangle b_i$
20:    end for
21:    Return $m(\lambda N, b_N)$
22: end procedure
```

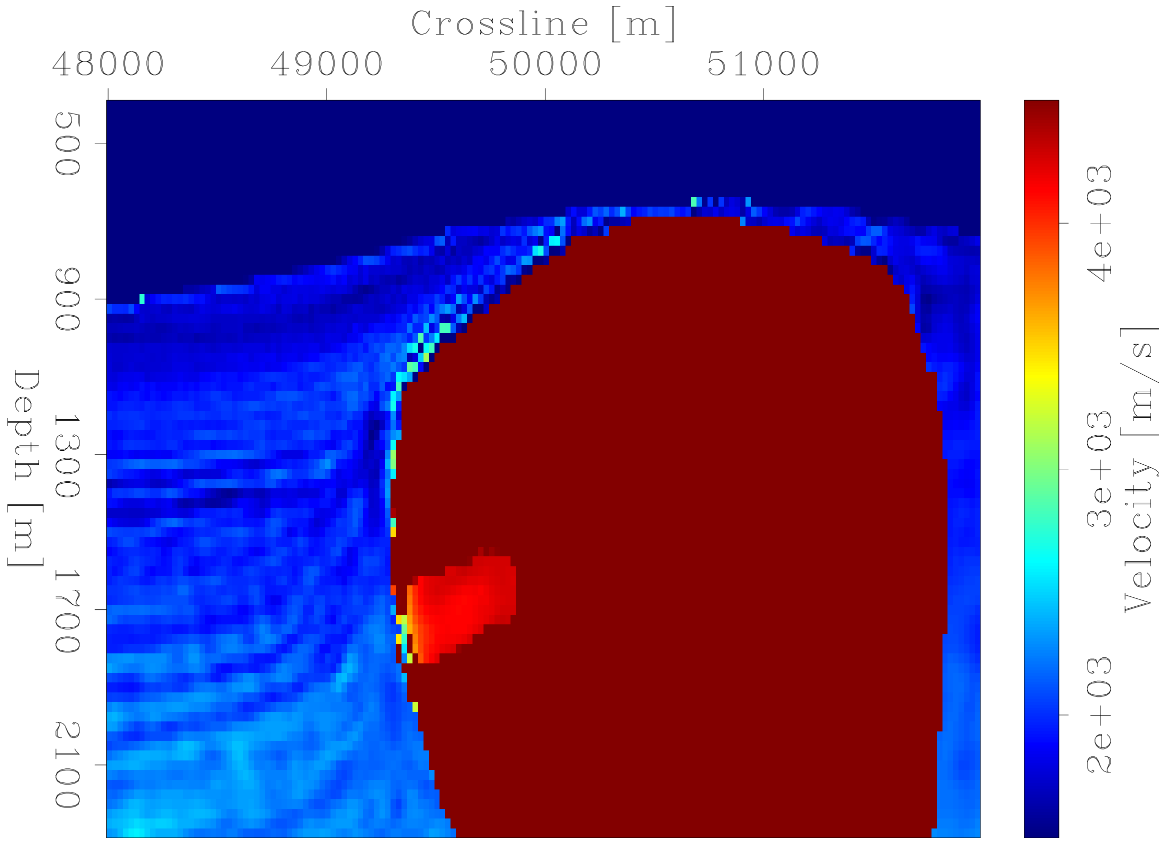


Figure 5.25: Velocity model after 35 iterations (214,800m inline position). chapter5/. velmodel-side-it-35

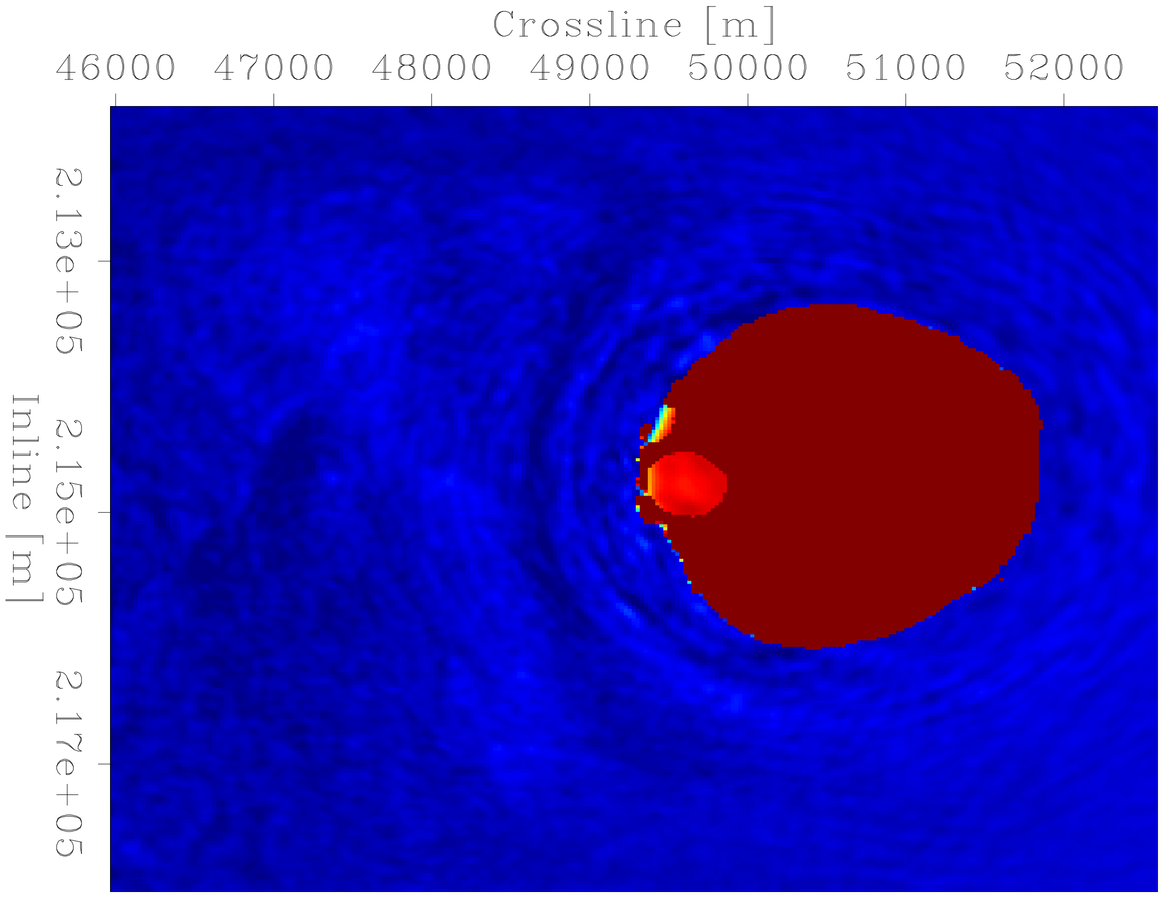


Figure 5.26: Velocity model after 35 iterations (1,700m depth position). chapter5/. velmodel-top-it-35

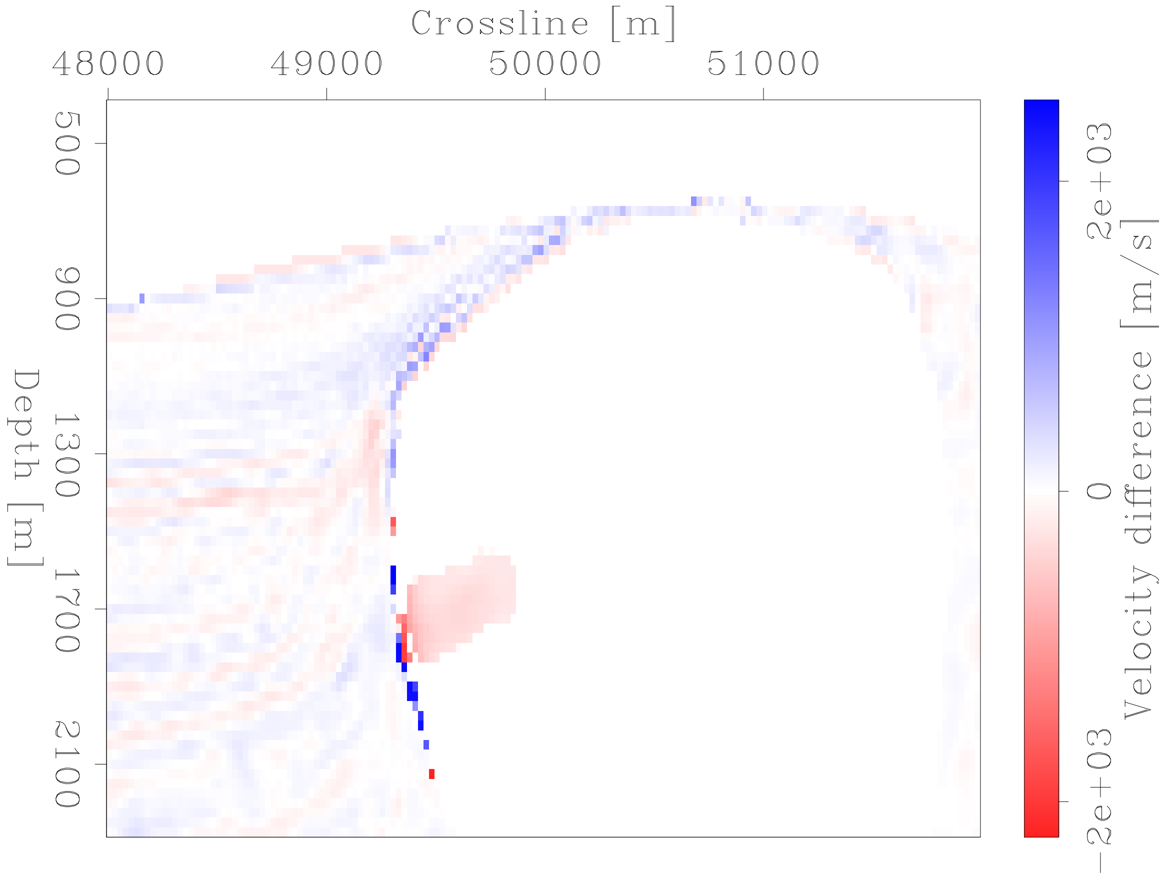


Figure 5.27: Full velocity model difference between model at 35 iterations and starting model (214,800m inline position). chapter5/. velmodel-diff-side-35

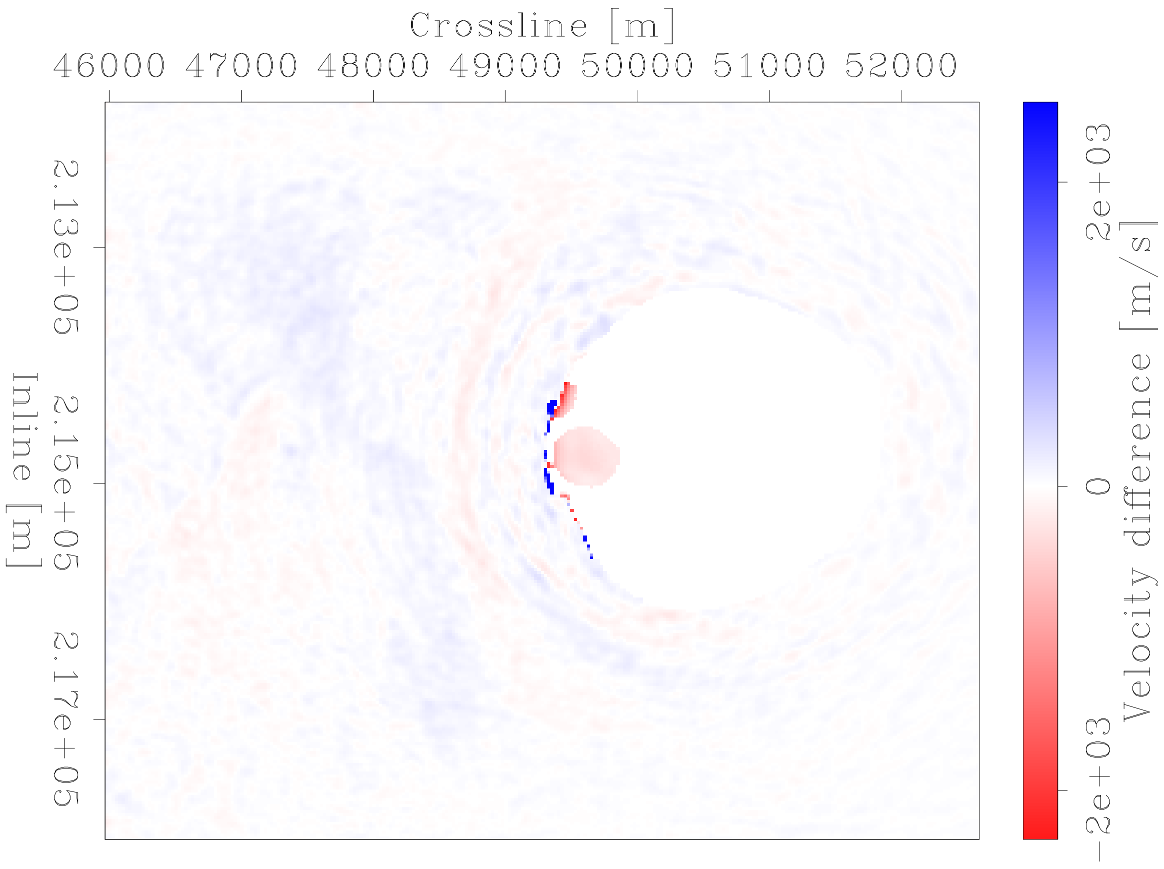


Figure 5.28: Full velocity model difference between model at 35 iterations and starting model (1,700m depth position). chapter5/. velmodel-diff-top-35

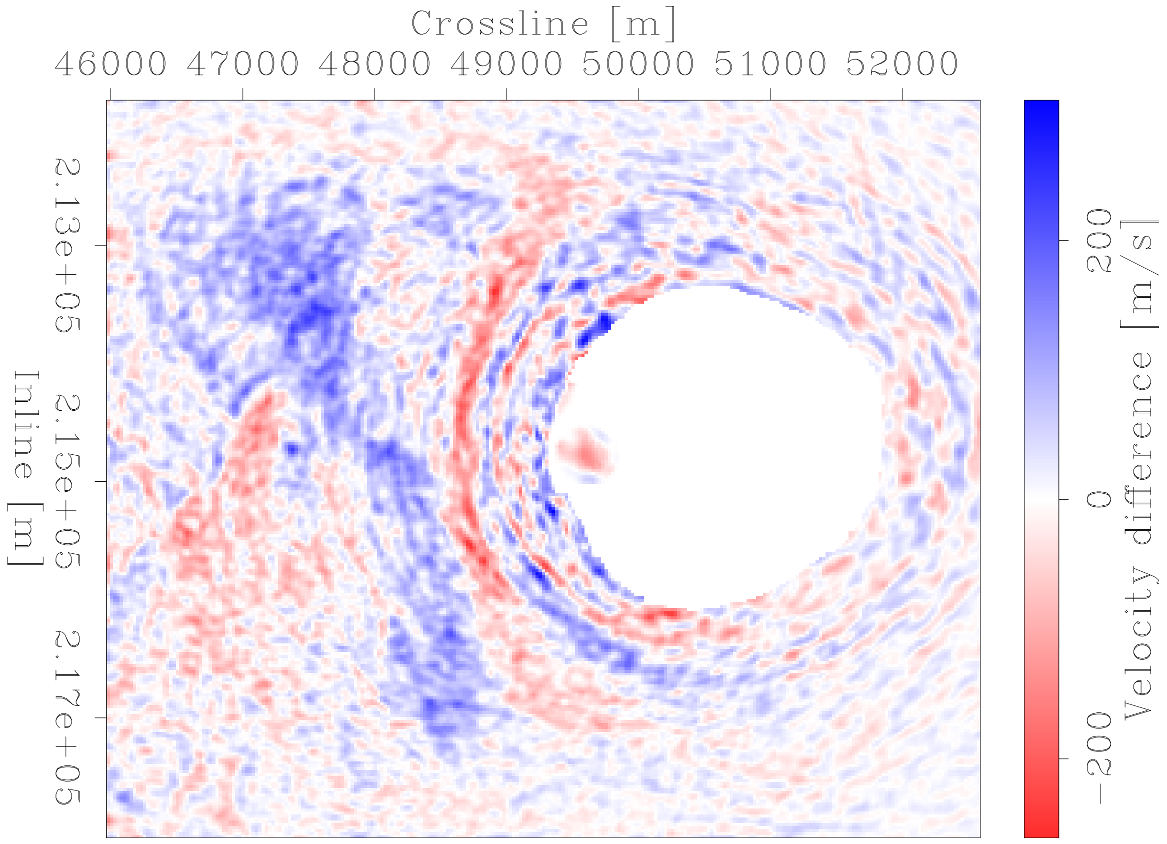


Figure 5.29: Background velocity model difference between model at 35 iterations and starting model (1,700m depth position). $\boxed{\text{chapter5/. velback-diff-top-35}}$

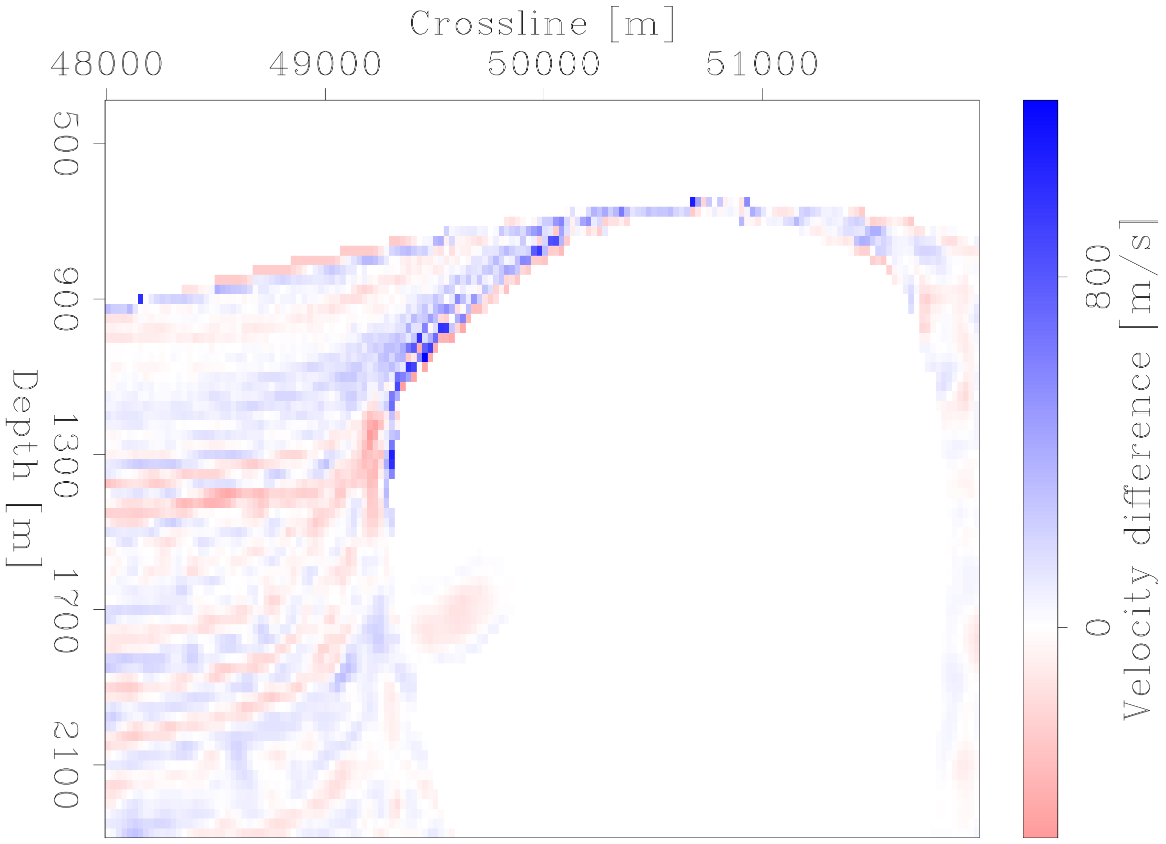


Figure 5.30: Background velocity model difference between model at 35 iterations and starting model (214,800m inline position). chapter5/. velback-diff-side-35

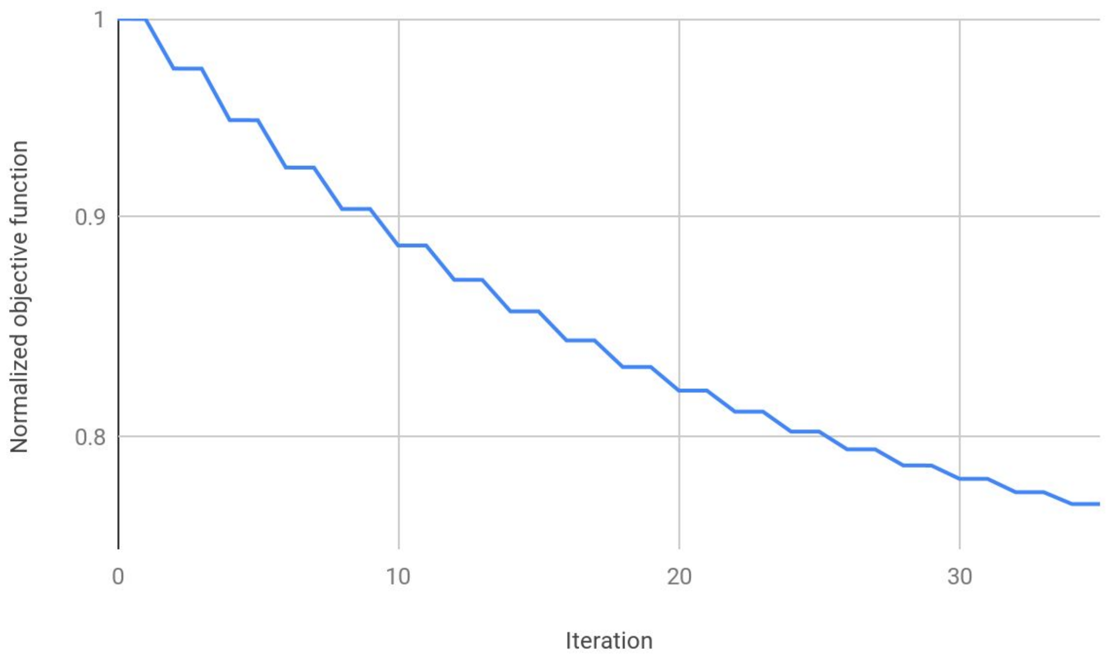


Figure 5.31: Objective function over non-linear (outer-loop) iterations. $\boxed{\text{chapter5/. inv-full-objfunc}}$

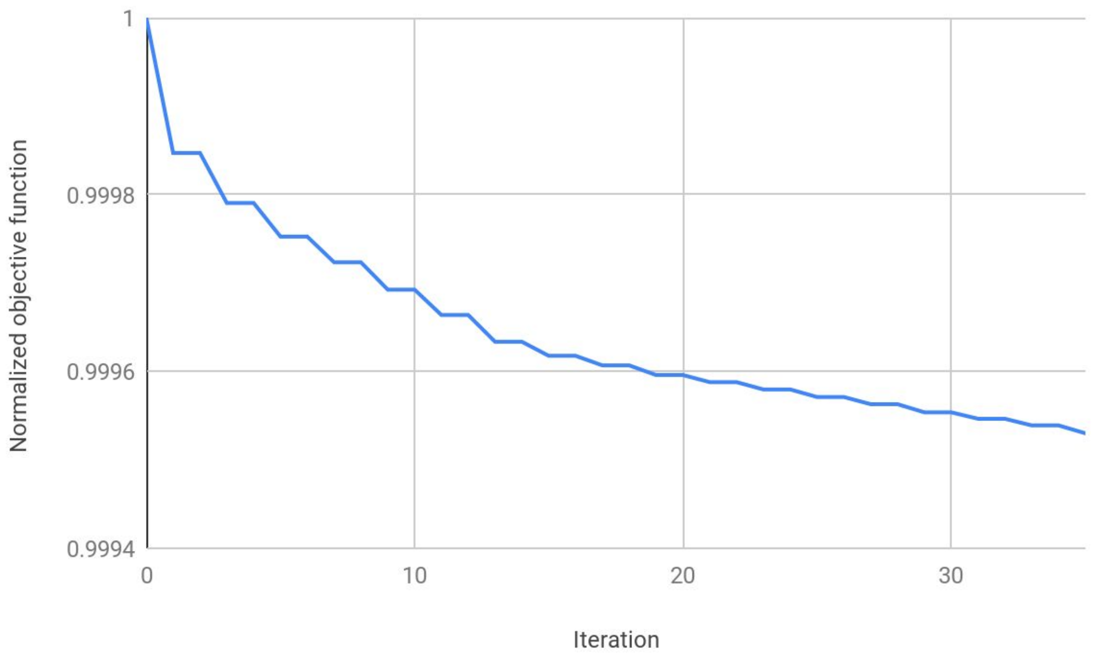


Figure 5.32: Contribution of level set updating to the objective function decrease over non-linear (outer-loop) iterations. $\boxed{\text{chapter5/. inv-salt-objfunc}}$

most difference would be the area directly below it. I find that coherency of some of the sediment layers is improved in the region where they meet the salt flank (compare Figures 5.34(a) and 5.34(b)). In order to determine how much influence the inclusion alone has on the image, I performed RTM with a model using the salt and inclusion updates, but with the original background velocity for areas outside the salt (Figure 5.35). I find that the image coherency is still improved on a number of reflectors below the inclusion. None of this level of detail is present in the original Shell RTM image (Figure 5.36). Similar improvements in the RTM image can be found below the inclusion at inline position 214,950m (compare Figures 5.38(a), 5.38(b), and 5.39).

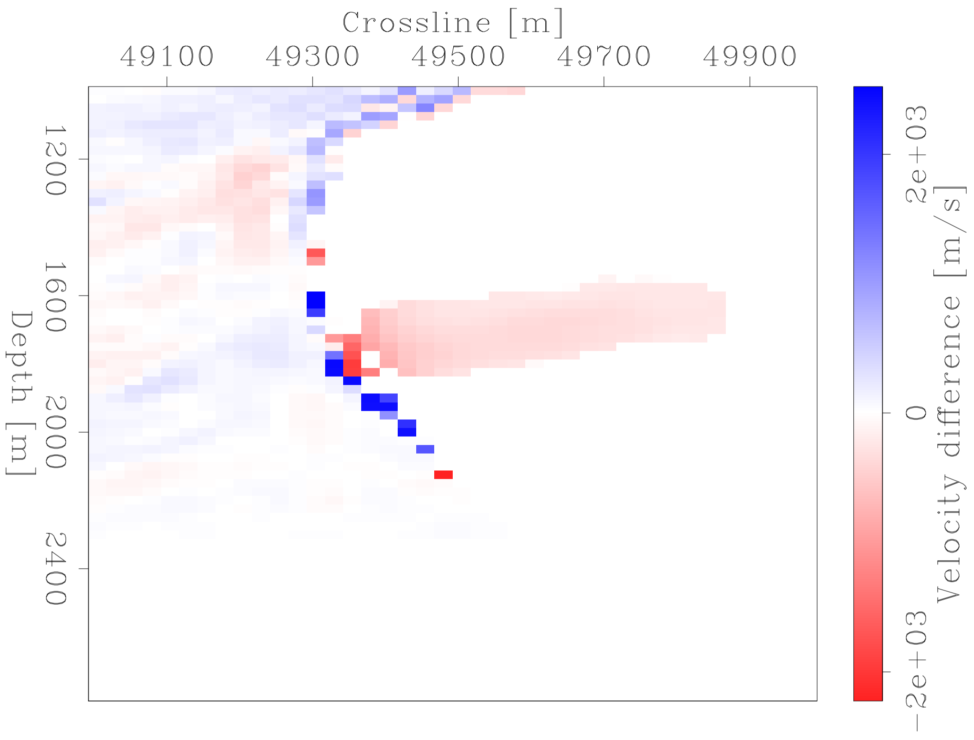
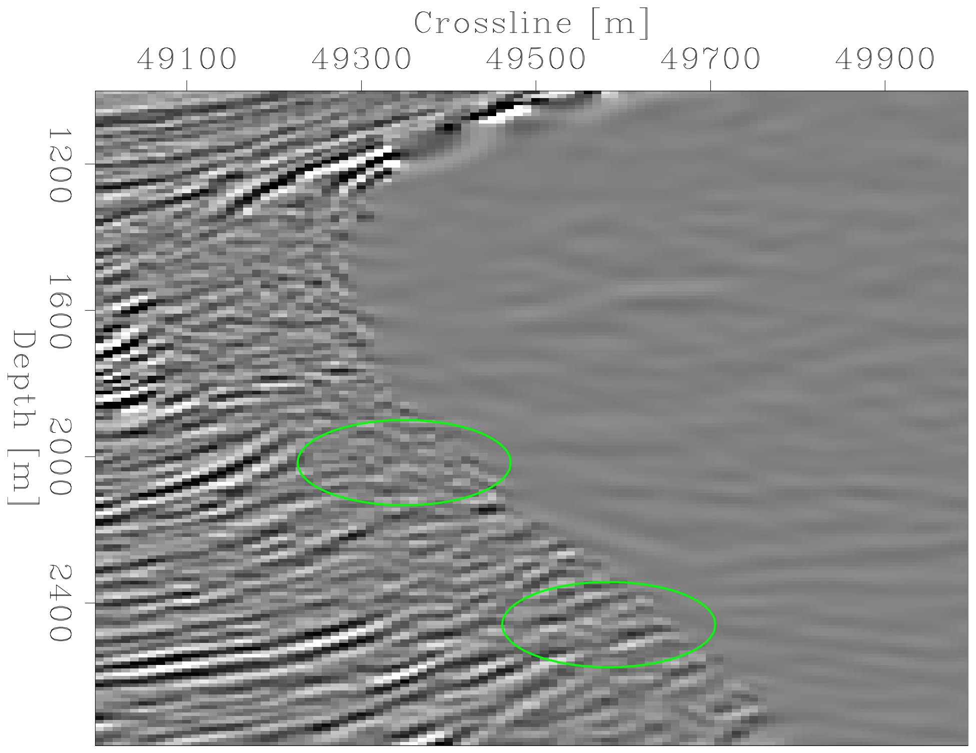
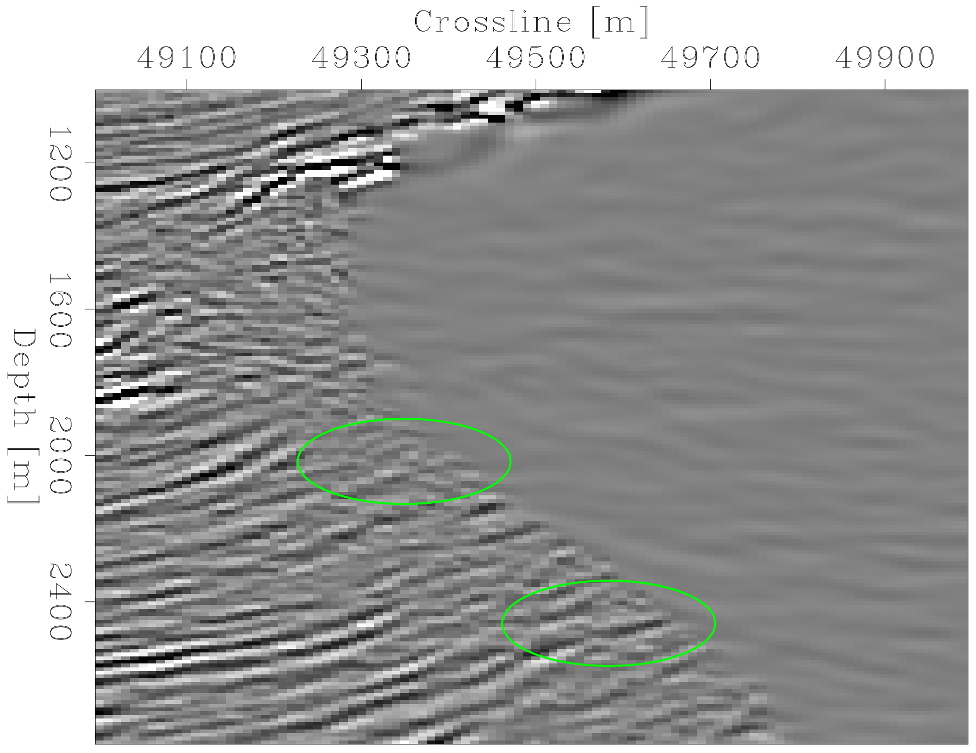


Figure 5.33: Difference between final velocity model and starting velocity model (214,800m inline position). chapter5/. inc-zone-vel-diff-214800



(a)



(b)

Figure 5.34: Migrated RTM image in inclusion zone at 214,800m inline position using starting velocity model (a) and with velocity model after 35 iterations (b). chapter5/. inc-zone-before-214800,inc-zone-after-214800

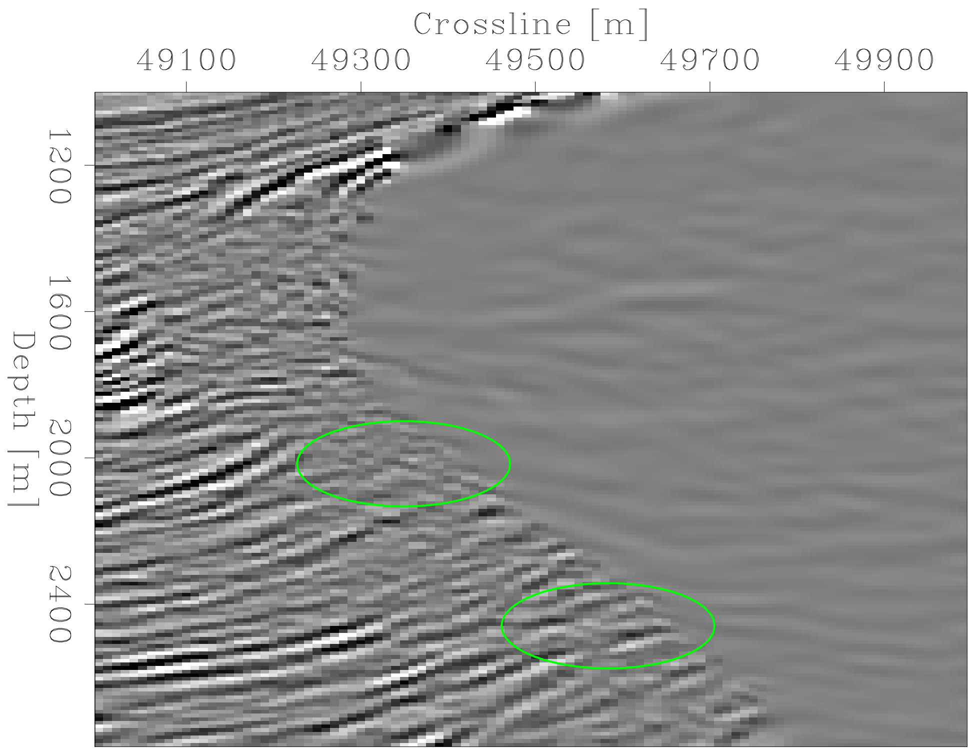


Figure 5.35: Migrated RTM image in inclusion zone using final salt model after 35 iterations (214,800m inline position), but using the original background velocity. chapter5/. inc-zone-after2-214800

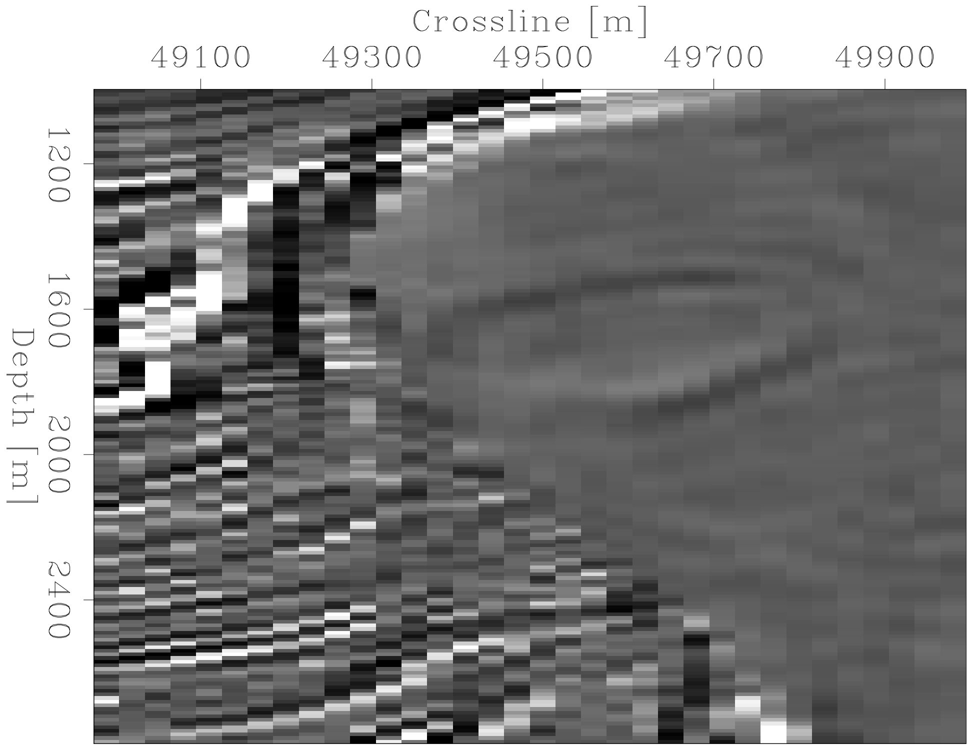


Figure 5.36: Migrated RTM image in inclusion zone provided by Shell (214,800m inline position). $\boxed{\text{chapter5/. inc-zone-shell-214800}}$

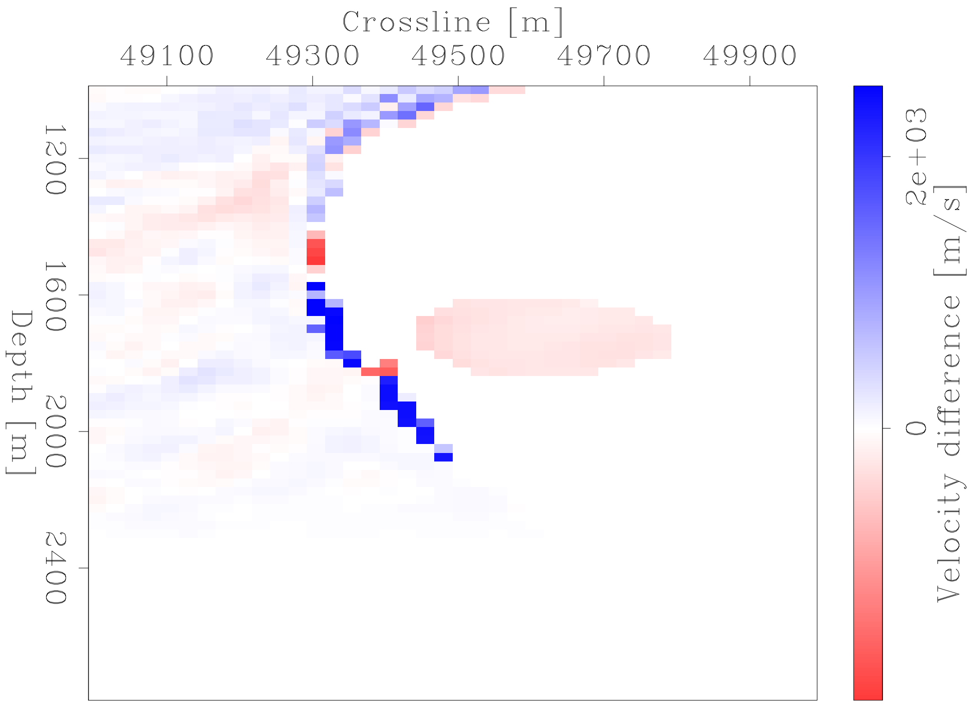
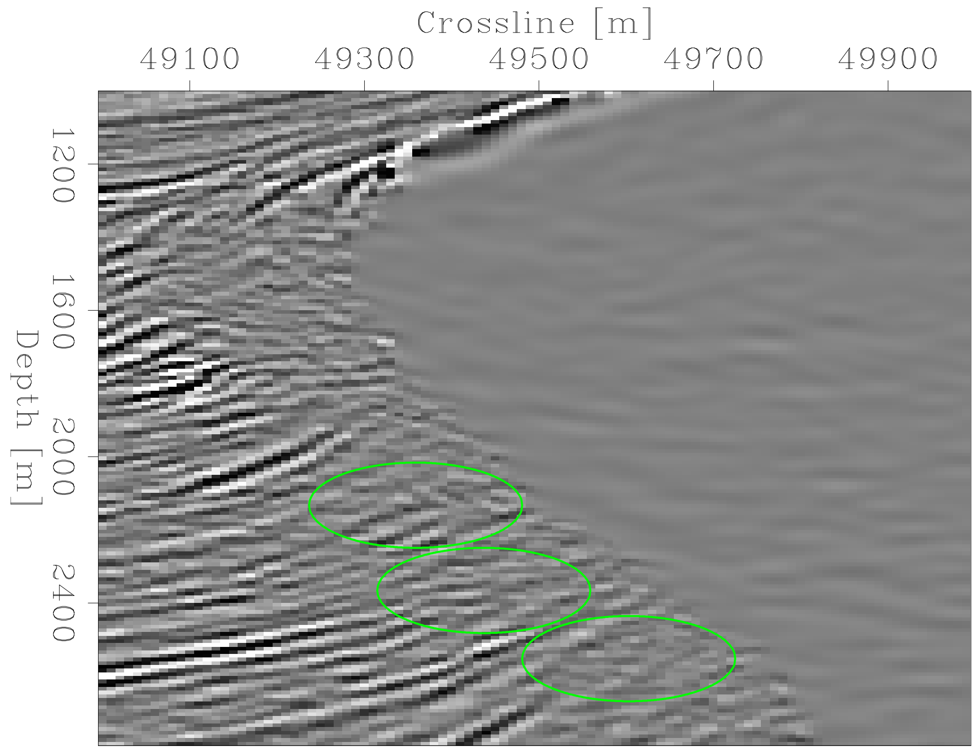


Figure 5.37: Difference between final velocity model and starting velocity model (214,950m inline position). chapter5/. inc-zone-vel-diff-214950

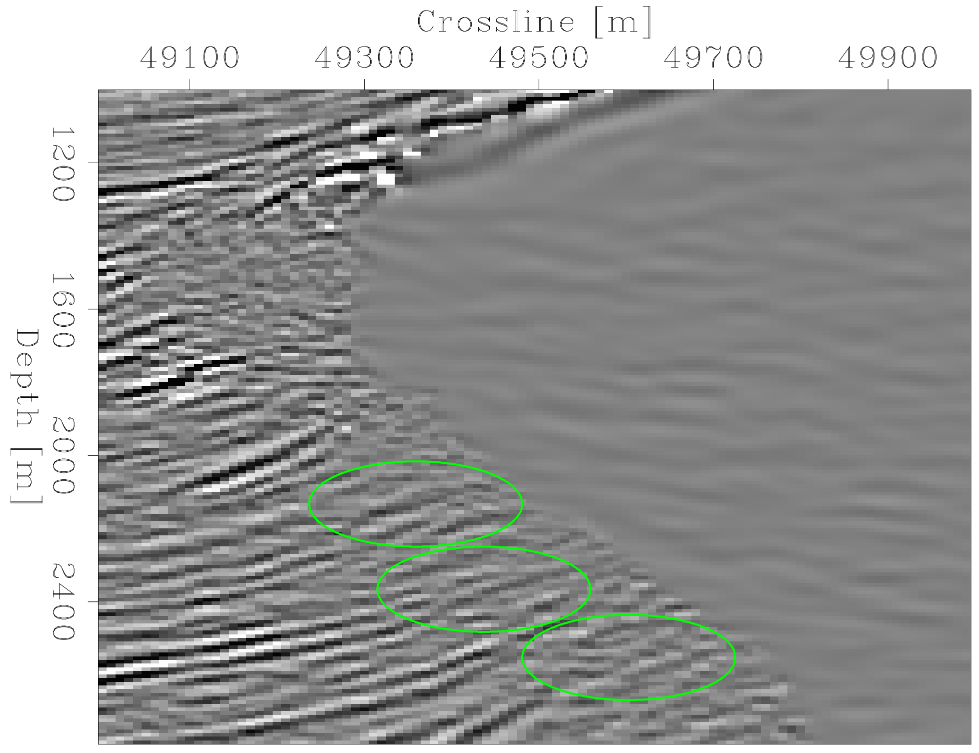
The area above the salt diapir is mostly improved by the background velocity model updating, but also seems to show a fault feature (see Figure 5.42(b)) that is not evident in the Shell RTM (Figure 5.41) or in the initial RTM image that I created (Figure 5.42(a)). A fault at this position could be produced from the upward stresses created by the salt diapir below.

CONCLUSIONS

In order to evaluate the effectiveness of shape optimization with level sets on a industry level data, I applied my method on a Gulf of Mexico OBN dataset from Shell. The data suggests an inclusion in the salt model, which makes it an ideal example for demonstrating the effectiveness of interpreter guidance as well. By intelligently selecting nodes that most effectively illuminate the inclusion area, I can temper the computational expense of the inversion. I explain how the phase-only objective function can alleviate the problems arising from amplitude differences between the field data and acoustically modeled data. I then invert for the level set shape and background velocity model using a Gauss-Newton Hessian inversion algorithm, and find



(a)



(b)

Figure 5.38: Migrated RTM image in inclusion zone at 214,950m inline position using starting velocity model (a) and with velocity model after 35 iterations (b). chapter5/. inc-zone-before-214950,inc-zone-after-214950

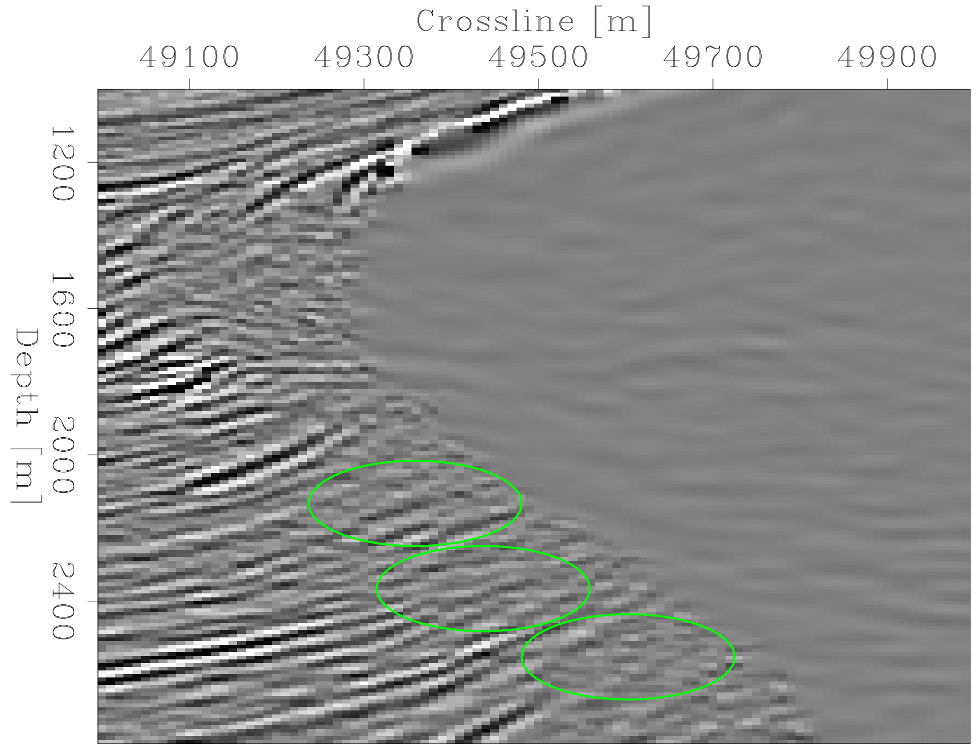


Figure 5.39: Migrated RTM image in inclusion zone using final salt model after 35 iterations (214,950m inline position), but using the original background velocity. chapter5/. inc-zone-after2-214950

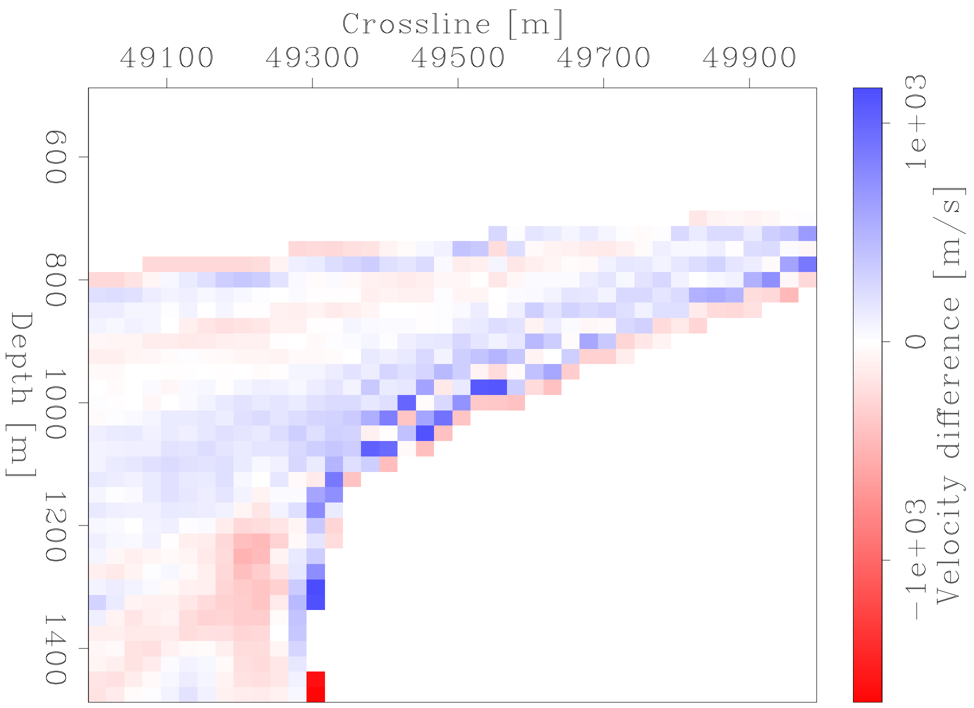


Figure 5.40: Difference between final velocity model and starting velocity model (214,820m inline position). chapter5/. top-zone-vel-diff-214820

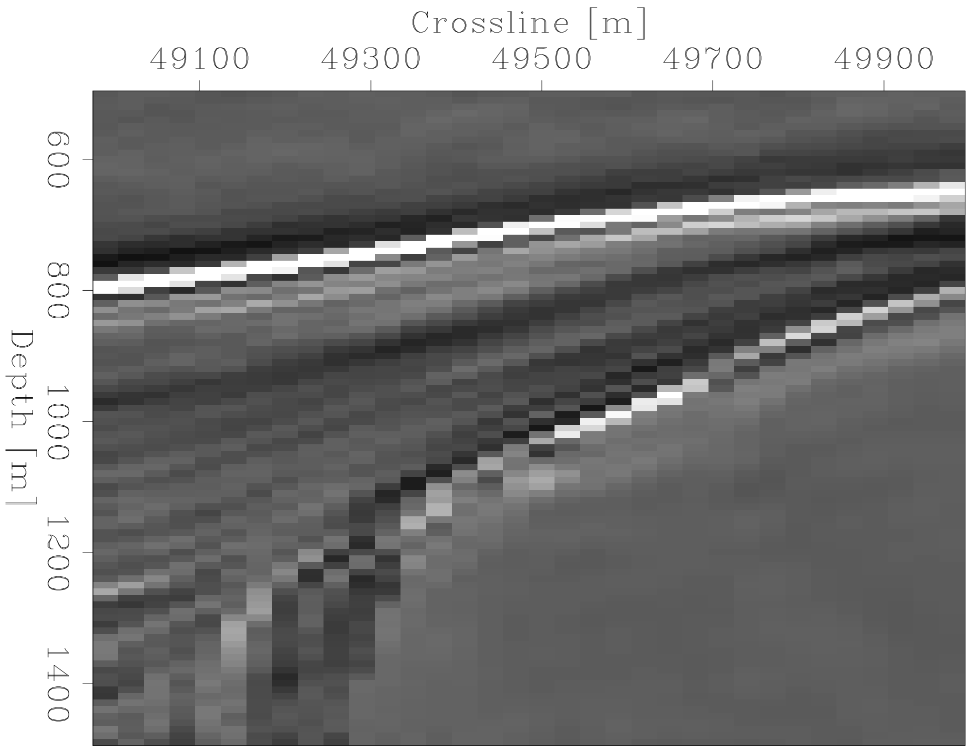
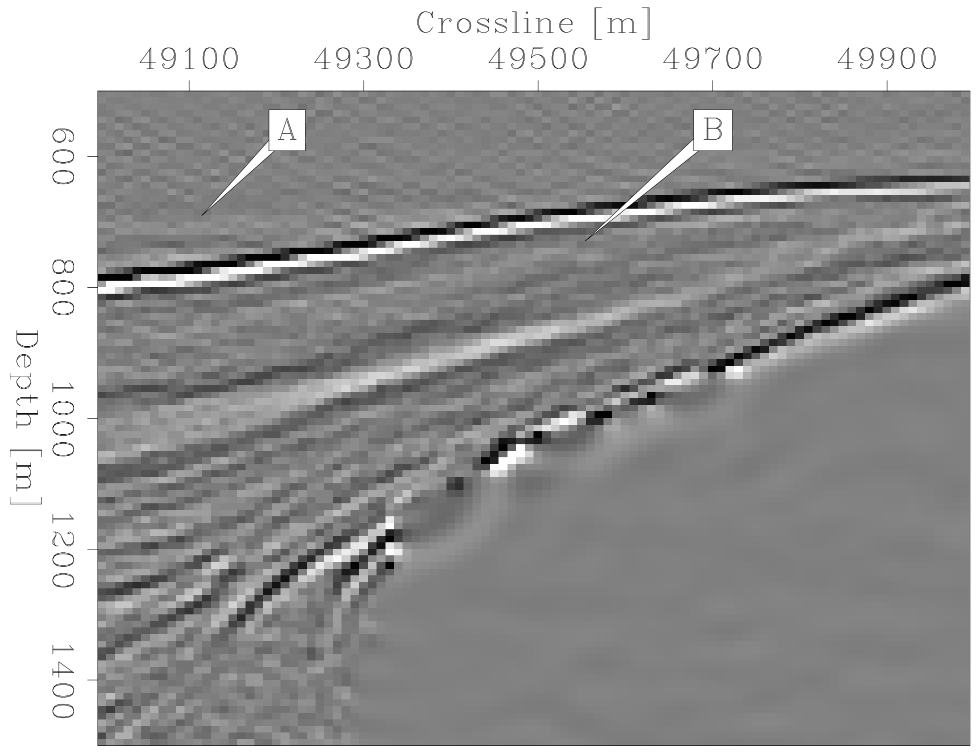
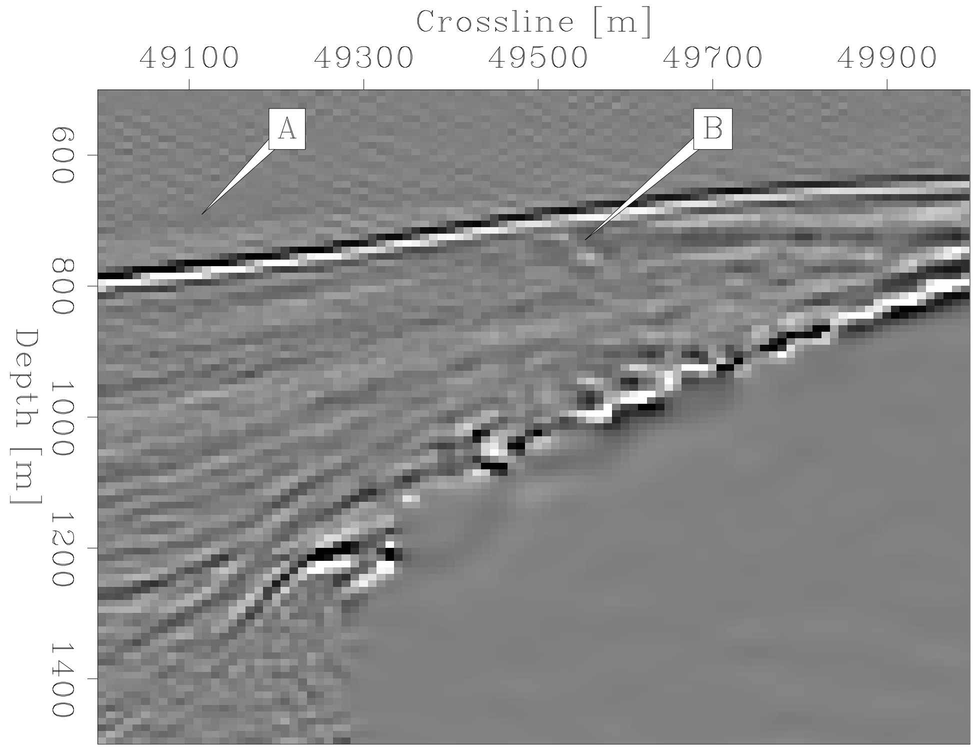


Figure 5.41: Migrated RTM image in top of salt zone provided by Shell (214,820m inline position). chapter5/. top-zone-shell-214820



(a)



(b)

Figure 5.42: Migrated RTM image in top of salt zone at 214,820m inline position using starting velocity model (a) and using velocity model after 35 iterations (b). (A) is the position of an artifact visible in Figure 5.42(a). (B) is the position of a potential fault, visible in (b). chapter5/. top-zone-before-214820,top-zone-after-214820

that it recovers an inclusion shape consistent with the RTM images. When I repeat the RTM images using the new velocity model, I find that I gain more coherency in the sediment layers directly below the inclusion that terminate against the salt flank, as well as other improvements near the top of salt and elsewhere. This demonstrates the efficacy of the method for application to field datasets.

Appendix A

Designaturing

FROM CIRCUITS TO RESPONSE CURVES

Hydrophones

The basic circuitry of a hydrophone has a capacitor (the peizo crystal that deforms under pressure) and the inherent resistance of the rest of the circuit that can be considered as a single resistor.

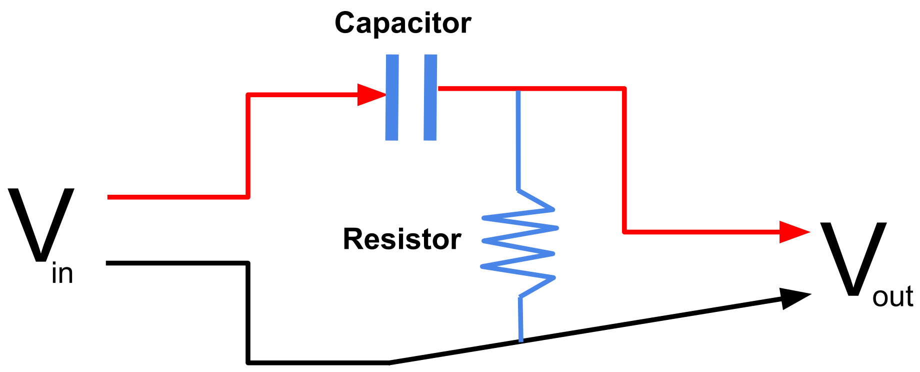


Figure A.1: Basic circuit diagram of a hydrophone. [**NR**] $\boxed{\text{appendix1/. hydro-circuit}}$

We can represent figure (A.1) with several equations that balance the current:

$$V_{in} - V_{out} = Q/C \tag{A.1}$$

$$V_{out} = IR \tag{A.2}$$

$$I = \frac{dQ}{dt} \tag{A.3}$$

Equation A.1 is the voltage on the capacitor, equation (A.2) is the current in the resistor, and equation (A.3) is the same current in the capacitor itself. What we are interested in is the output voltage V_{out} and how it relates to the input voltage for different frequencies:

$$V_{out} = f(\omega, RC)V_{in} \tag{A.4}$$

$$\tag{A.5}$$

To derive this formulation, we begin by taking the time derivative of equation (A.1) in order to eliminate Q:

$$\begin{aligned}(V_{in} - V_{out}) &= Q/C \\ \frac{d}{dt}(V_{in} - V_{out}) &= \frac{dQ}{dt}/C,\end{aligned}$$

and then combine with equation (A.3):

$$\frac{d}{dt}(V_{in} - V_{out}) = I/C,$$

To eliminate I, we substitute in $I = V_{out}/R$ from equation (A.2):

$$\frac{d}{dt}(V_{in} - V_{out}) = V_{out}/RC$$

To calculate V_{in} from V_{out}, move all V_{out} to the right side:

$$\frac{dV_{in}}{dt} - \frac{dV_{out}}{dt} = V_{out}/RC \tag{A.6}$$

$$\frac{dV_{in}}{dt} = \frac{dV_{out}}{dt} + V_{out}/RC \tag{A.7}$$

We can solve equation A.7 in the fourier space by making the substitution $\frac{d}{dt} = i\omega$:

$$\begin{aligned} i\omega V_{in} &= i\omega V_{out} + V_{out}/RC \\ V_{in} &= V_{out} + \frac{V_{out}}{i\omega RC} \\ V_{in} &= V_{out}\left(1 + \frac{1}{i\omega RC}\right) \\ i\omega RC V_{in} &= V_{out}\left(i\omega RC + 1\right) \end{aligned}$$

This means that in Fourier space we can define our transfer function as:

$$f(\omega, RC) = \frac{i\omega RC}{(i\omega RC + 1)} \tag{A.8}$$

from which we can find the original input data by solving equation (A.8) for V_{in}. From this function, we can easily find the amplitude scaling and the phase shifting that the instrument creates in the recorded data:

$$\phi(\omega) = \tan^{-1}\left[\frac{\mathrm{real} f(\omega, RC)}{\mathrm{imag} f(\omega, RC)}\right] \tag{A.9}$$

$$\tag{A.10}$$

Alternatively, if we aren't interested in actually analyzing the transfer function, but just want to deconvolve our instrument response from our recorded data, we can solve for V_{in} from equation A.7 using a finite difference approach:

$$\frac{V_{in}[i] - V_{in}[i-1]}{dt} = \frac{V_{out}[i] - V_{out}[i-1]}{dt} + \frac{V_{out}[i] + V_{out}[i-1]}{2RC}$$

$$V_{in}[i] = V_{in}[i-1] + V_{out}[i] - V_{out}[i-1] + \frac{(V_{out}[i] + V_{out}[i-1])dt}{2RC}$$

Geophones

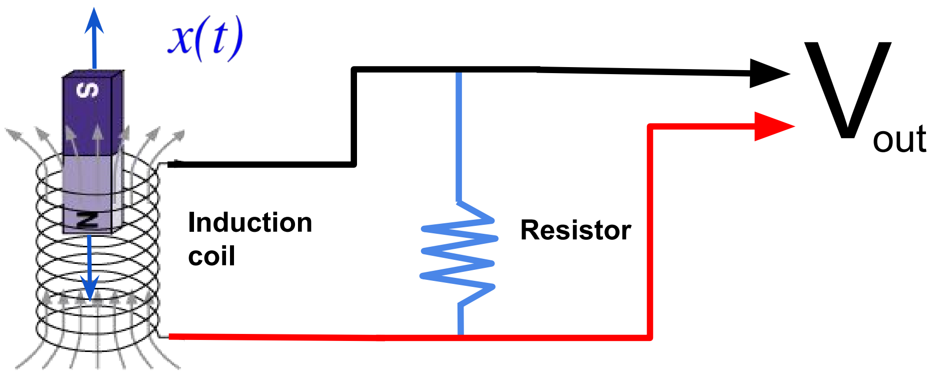


Figure A.2: Basic circuit diagram of a geophone. [**NR**] appendix1/. geophone-circuit-0

For geophones, instead of having a piezo crystal modulate an input voltage, we have a mechanical system that creates voltage by converting motion of a magnet through a coil.

The motion $x(t)$ that occurs in the coil produces a voltage, and of course the circuit has some general resistance to account for. However, the most significant part of the geophone response curve comes from the equations of motion for the magnet relative to the coil.

Figure A.3 shows the internal and external forces at play in a vertical component geophone, and Figure A.4 shows the movement of the magnet relative to the instrument frame. We can begin our derivation of the instrument response by writing a force balance equation to balance the external (left hand side) and internal forces (right hand side).

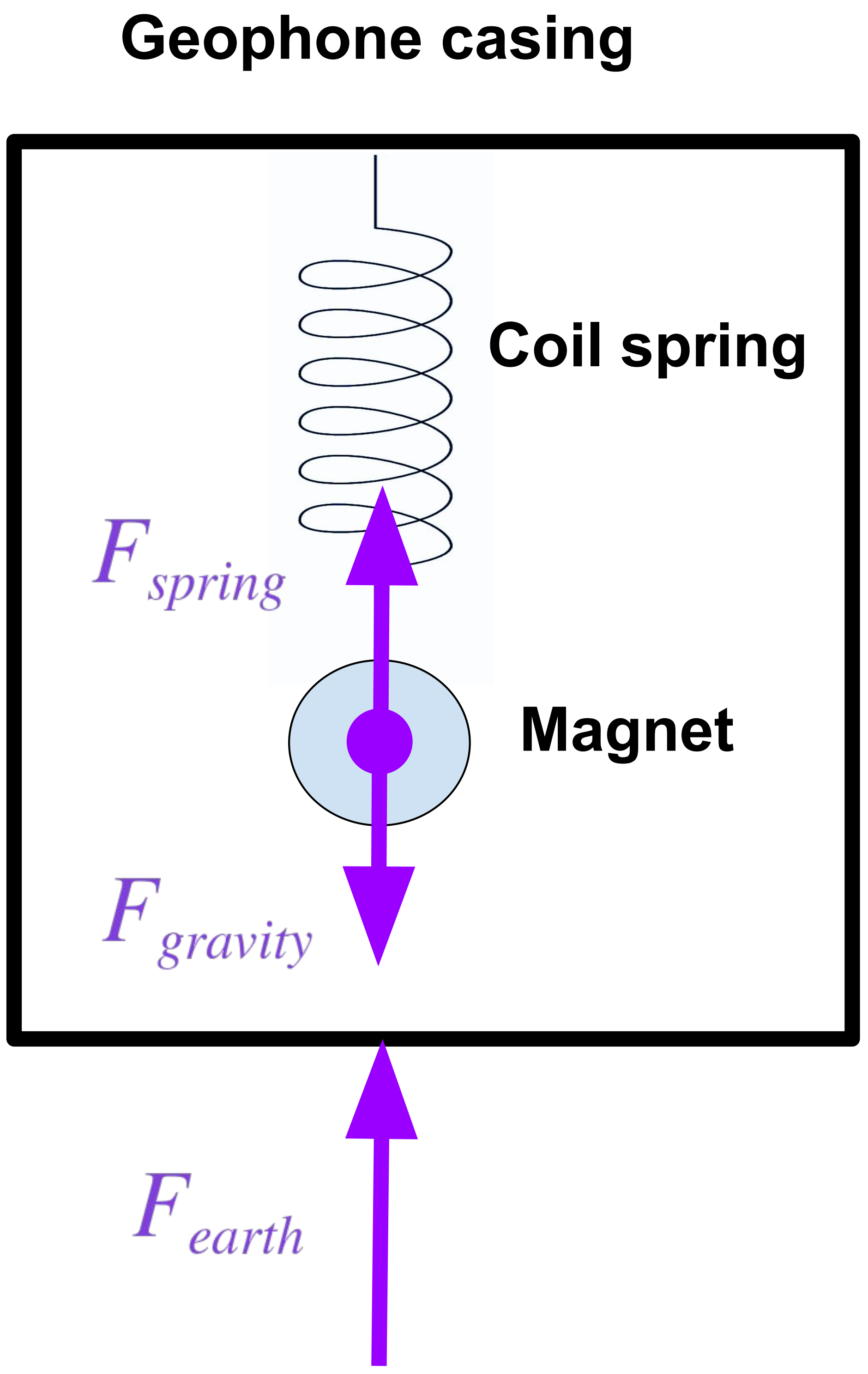


Figure A.3: Diagram of internal and external static forces. [**NR**] appendix1/. geophone-physics

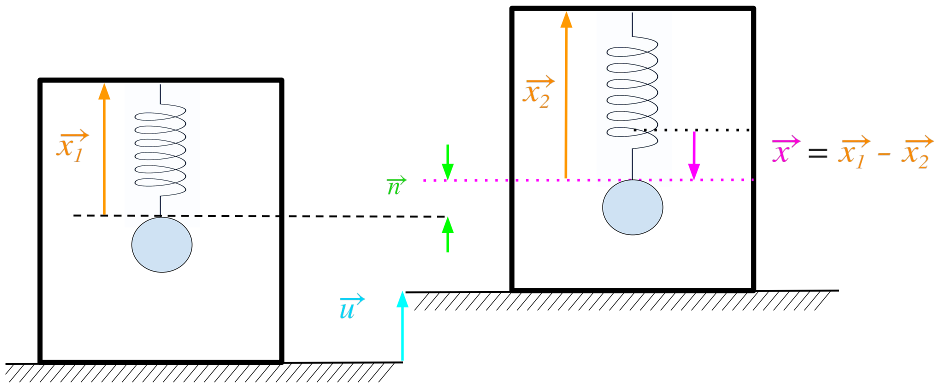


Figure A.4: Diagram of vertical component geophone responding to ground motion. **u** is the motion of the earth, and **x** is the motion of the magnet relative to the coil. [**NR**] $\boxed{\text{appendix1/. geophone-physics2}}$

$$m\frac{\partial^2}{\partial t^2}(u+x) = -kx \tag{A.11}$$

Where m is the mass of the magnet, and k is the spring constant. We can re-define k according the natural frequency of the spring and mass system:

$$k = \omega_0^2 m,$$

which gives us the equation of harmonic motion:

$$\frac{\partial^2 x}{\partial t^2} + \omega_0^2 x = -\frac{\partial^2 u}{\partial t^2} \tag{A.12}$$

Equation A.12 defines an undamped system, which means it would be overcome by frequencies close to the natural frequency of the system (ω_0). In order to allow for the recording of other frequencies, we add a dampening term that is proportional to the velocity of the magnet:

$$\frac{\partial^2 x}{\partial t^2} + \omega_0^2 x + 2\omega_0\lambda\frac{\partial x}{\partial t} = -\frac{\partial^2 u}{\partial t^2}, \tag{A.13}$$

where λ is the dampening ratio. A simple way to solve equation A.13 is to use the Fourier transform, which allows us to subsitute $i\omega$ with $\frac{\partial}{\partial t}$. Since the actual measurement that we record is voltage, and voltage is a function of the velocity of the magnet, we ultimately would like to solve for $\frac{\partial x}{\partial t}$:

$$\begin{aligned}\frac{\partial}{\partial t}\frac{\partial x}{\partial t} + \omega_0^2 x + 2\omega_0\lambda\frac{\partial x}{\partial t} &= -\frac{\partial^2 u}{\partial t^2}\\ i\omega\frac{\partial x}{\partial t} - \frac{\omega_0^2}{i\omega}\frac{\partial x}{\partial t} + 2\omega_0\lambda\frac{\partial x}{\partial t} &= \omega^2 u\end{aligned}$$

This means we can relate the output voltage ($\frac{\partial x}{\partial t}$) with the input acceleration of the earth ($\frac{\partial^2 u}{\partial t^2}$):

$$\frac{\frac{\partial x}{\partial t}}{\frac{\partial^2 u}{\partial t^2}} = \frac{-i\omega}{-\omega^2 + \omega_0^2 + i2\omega\omega_0\lambda} \tag{A.14}$$

DE-SIGNATURING DATA

Application to Cardamom data

In order to implement equations A.9 and A.14, we perform the division in the Fourier domain. Our algorithm works according to the following workflow:

Algorithm 3 Remove instrument response from data

```
1: procedure ResponseRemoval($data$,$response$)
2:     for each trace $i$ in $data$ do
3:         $temp(i) = \text{FFT}(data(i))$
4:         $temp(i).amp = \sqrt{(temp(i).imag)^2 + (temp(i).real)^2}$
5:         $temp(i).phase = \tan^{-1}\left(\frac{temp(i).imag}{temp(i).real}\right)$
6:         $temp2(i).phase = temp(i).phase - response.phase$
7:         $temp2(i).amp = \frac{temp(i).amp}{response.amp}$
8:         $output(i) = \text{FFT}^{-1}(temp2(i))$
9:     end for
10:    Return $output$
11: end procedure
```

We perform the actual instrument response removal in the Fourier domain, and then convert back to the time domain. The response curves are functions of frequency as we describe in equations A.13 and A.14. We can see the impact of the response removal in Figure A.5 and Figure A.6. Further, the difference in the spectra can be seen clearly in Figures A.7 and A.8, where the lower frequencies are noticably boosted.

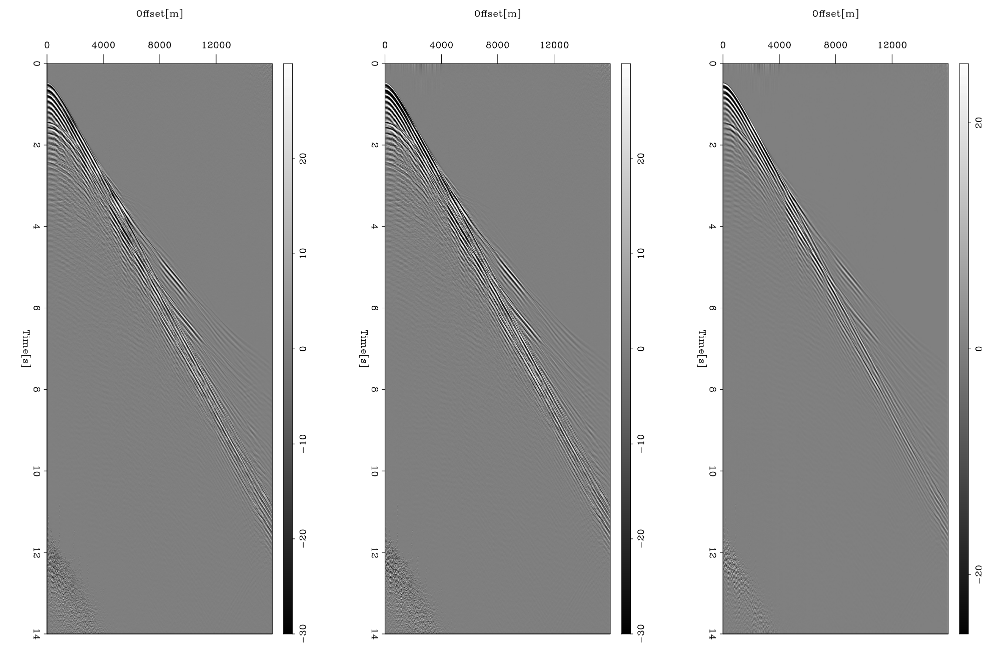


Figure A.5: Hydrophone data original (left), designatured (middle) and difference (right). Clipped to emphasize differences. [**ER**] appendix1/. hydro-response-removal-comp

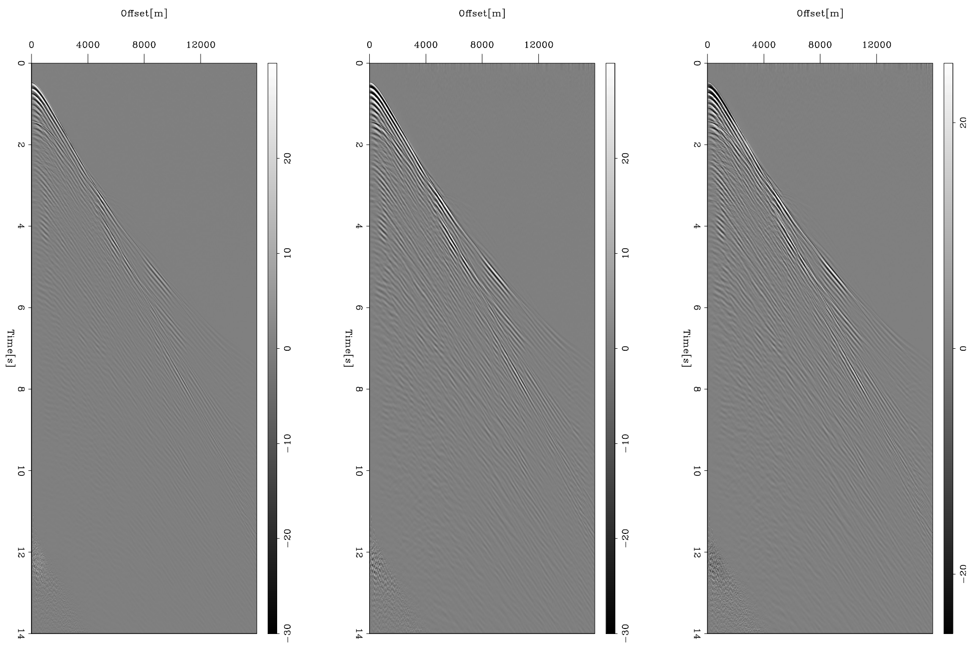


Figure A.6: Geophone data original (left), designatured (middle) and difference (right). Clipped to emphasize differences. [**ER**] appendix1/. geophone-response-removal-comp

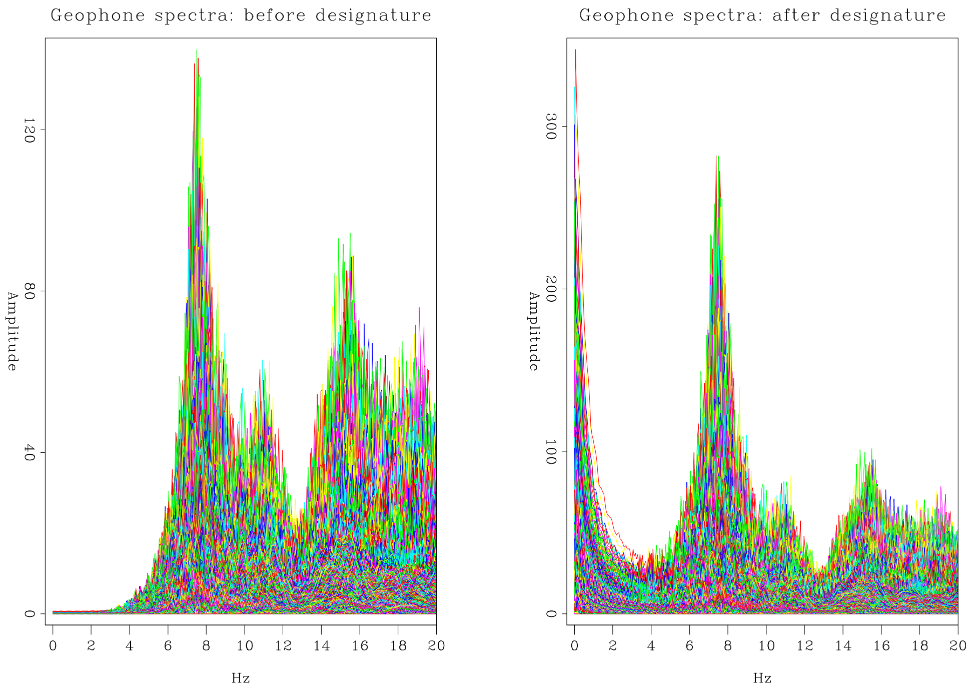


Figure A.7: Geophone data spectra before response removal (left), and after (right) for each trace. [**ER**] appendix1/. spectra-vert-compare

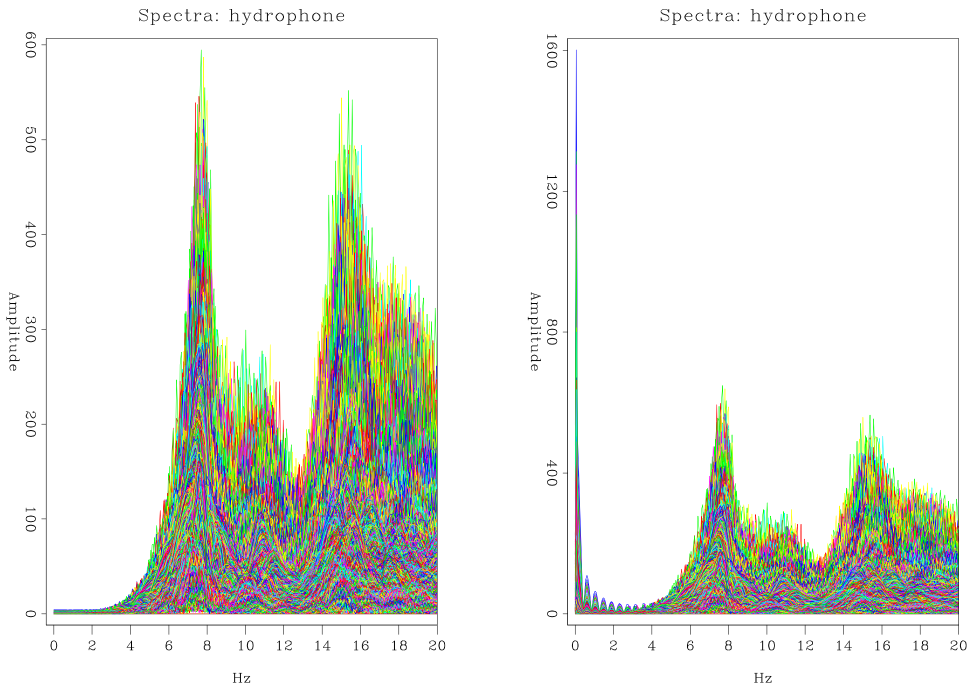


Figure A.8: Hydrophone data spectra before response removal (left), and after (right) for each trace. [**ER**] $\boxed{\text{appendix1/. spectra-hydr-compare}}$

Appendix B

Data Pre-processing

In order to successfully perform an FWI style of inversion with our dataset, we first need to apply a standard processing flow. An important goal to keep in mind is that we ultimately want to calculate a 'good' residual for use in our inversion. To do this, we need to have some similarity between our synthetically modeled data and the observed data. One feature in our observed data that can be difficult to accurately recreate in our synthetic data is the bubble that follows the initial source injection. For this reason, we choose to remove it from the observed data.

This appendix intends to show that a straight-forward way to simultaneously remove the bubble and increase similarity between our observed and synthetic data is to shape the observed data to the source wavelet we use in our synthetic modeling. To do this, we first perform separation of the up and downgoing components from the hydrophone data using PZ-summation. Next, we extract a representative wavelet from the observed data first-arrival, and then estimate a filter that shapes it to the synthetic data source wavelet. We then apply this filter to the entire dataset, resulting in a debubbled dataset that has a high degree of phase similarity with our synthetic data.

PZ-summation

Since the dataset was recorded on ocean bottom nodes (OBN), the first arrival in the hydrophone component contains an up-going ocean bottom reflection that nearly coincides with the downgoing direct arrival event. This means the wavelet we extract from the first arrival event will have an ocean bottom event mixed into it. To get an accurate estimated source wavelet, we need to isolate the downgoing direct arrival. For this reason, we first need to perform PZ-summation before we do any source wavelet estimation.

PZ-summation is a technique used to separate the up and down going components of hydrophone data (figure B.1) using the complimentary information found in the vertical component data (figure B.2). We base our application of this processing step on the approach and assumptions used by Biondi and Levin (2014) (originally based on Melbo et al. (2002)), which represents the up and downgoing data with the following equations:

$$P_{up}(f,k) = \frac{1}{2}P(f,k) + a(f)\frac{\rho}{2q(f,k)}Z(f,k), \tag{B.1}$$

$$P_{down}(f,k) = \frac{1}{2}P(f,k) - a(f)\frac{\rho}{2q(f,k)}Z(f,k), \tag{B.2}$$

where P is the designatured pressure data, Z is the designatured vertical data, $a(f)$ is the calibration filter, ρ is the water density, and q is the vertical slowness of the water layer defined as:

$$q(f,k) = \sqrt{c^{-2} - p^2(f,k)}. \tag{B.3}$$

In this case, c is the water velocity at the receiver position and p is the ray parameter.

One event in the data we can leverage is the refraction event, which by definition is an upgoing event. Furthermore, the refraction event is naturally separated in the time domain at far offsets, making it easy to isolate (see figures B.3 and B.4).

We can estimate a filter $\hat{a}(f)$ to apply to the windowed refraction event such that it minimizes its energy. However, the inverted $\hat{a}(f)$ contains the effect of the vertical slowness and water density. We can represent this with equation B.4:

$$\hat{a}(f) = a(f)\frac{\rho}{2q(f,k)}. \tag{B.4}$$

However, we assume the water velocity and density are constant throughout, and

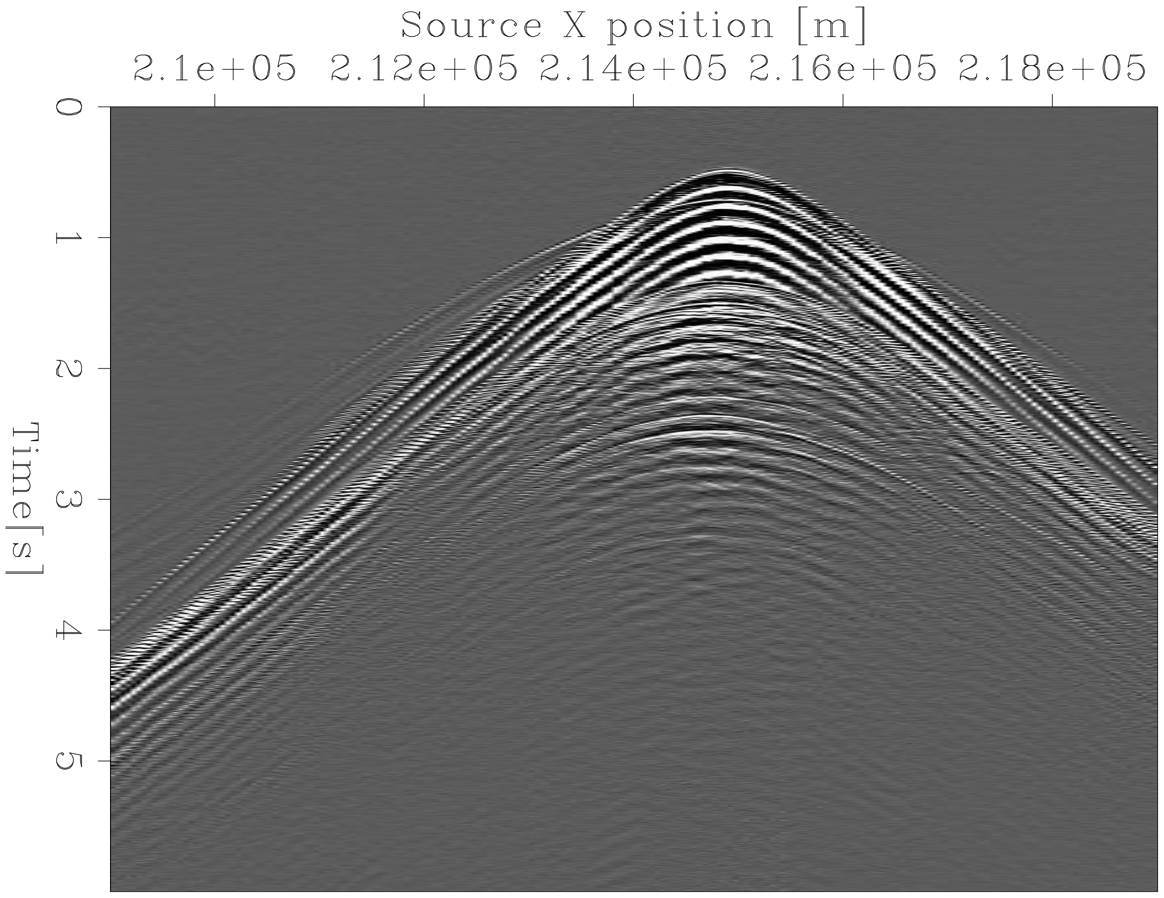


Figure B.1: 2D line receiver gather example from designatured and bandpassed hydrophone data. [**CR**] $\boxed{\text{appendix2/. inputHydro}}$

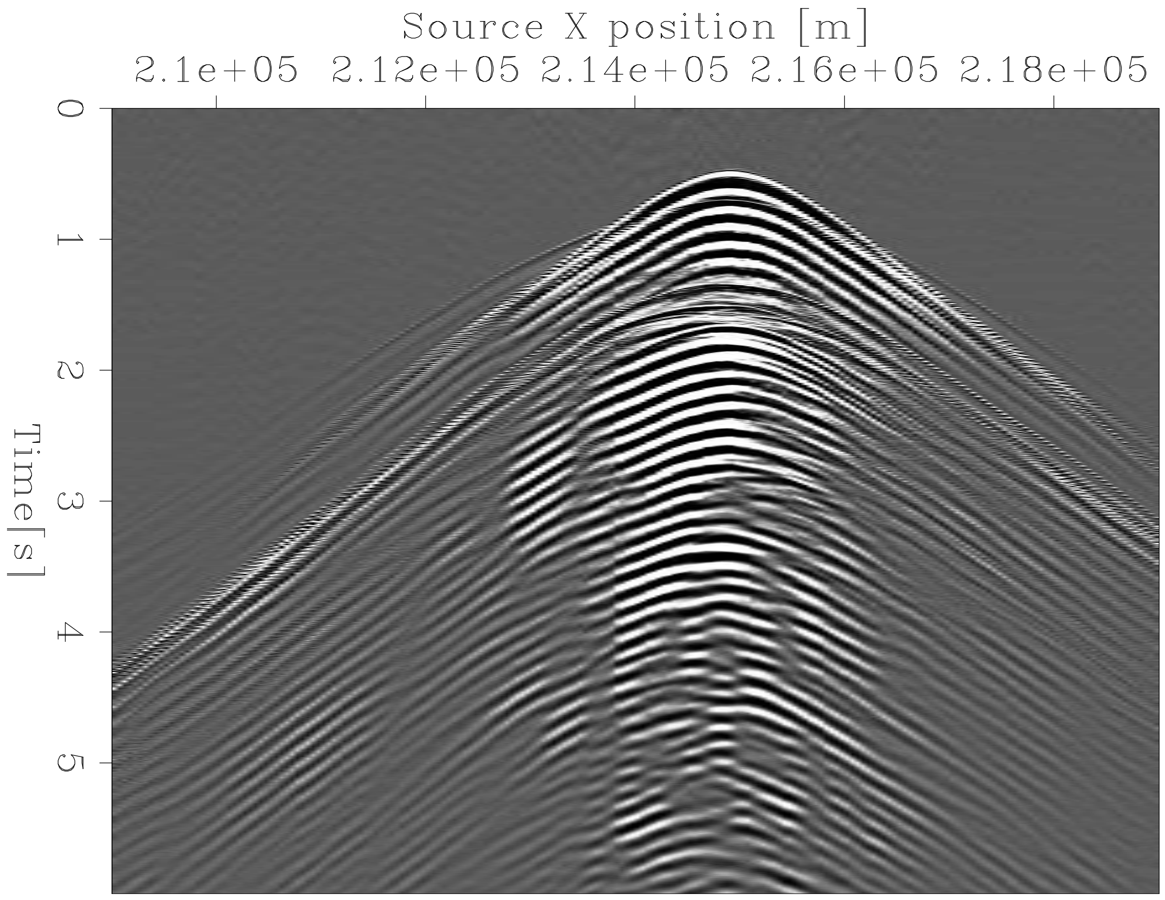


Figure B.2: 2D line receiver gather example from designatured and bandpassed geophone data. [**CR**] appendix2/. inputGeo

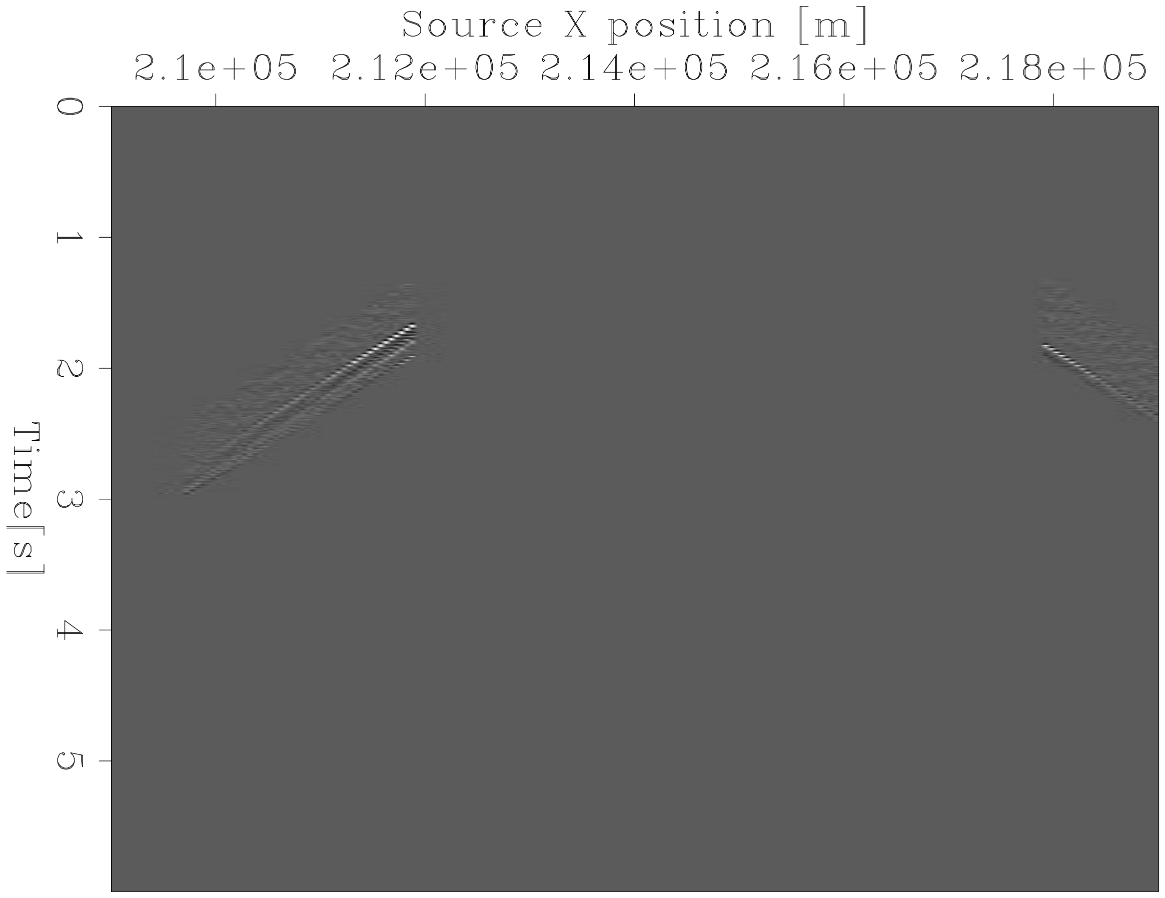


Figure B.3: Hydrophone data window (isolating the refraction events) used for estimating PZ-summation filter. [**CR**] $\boxed{\text{appendix2/. Pwindow}}$

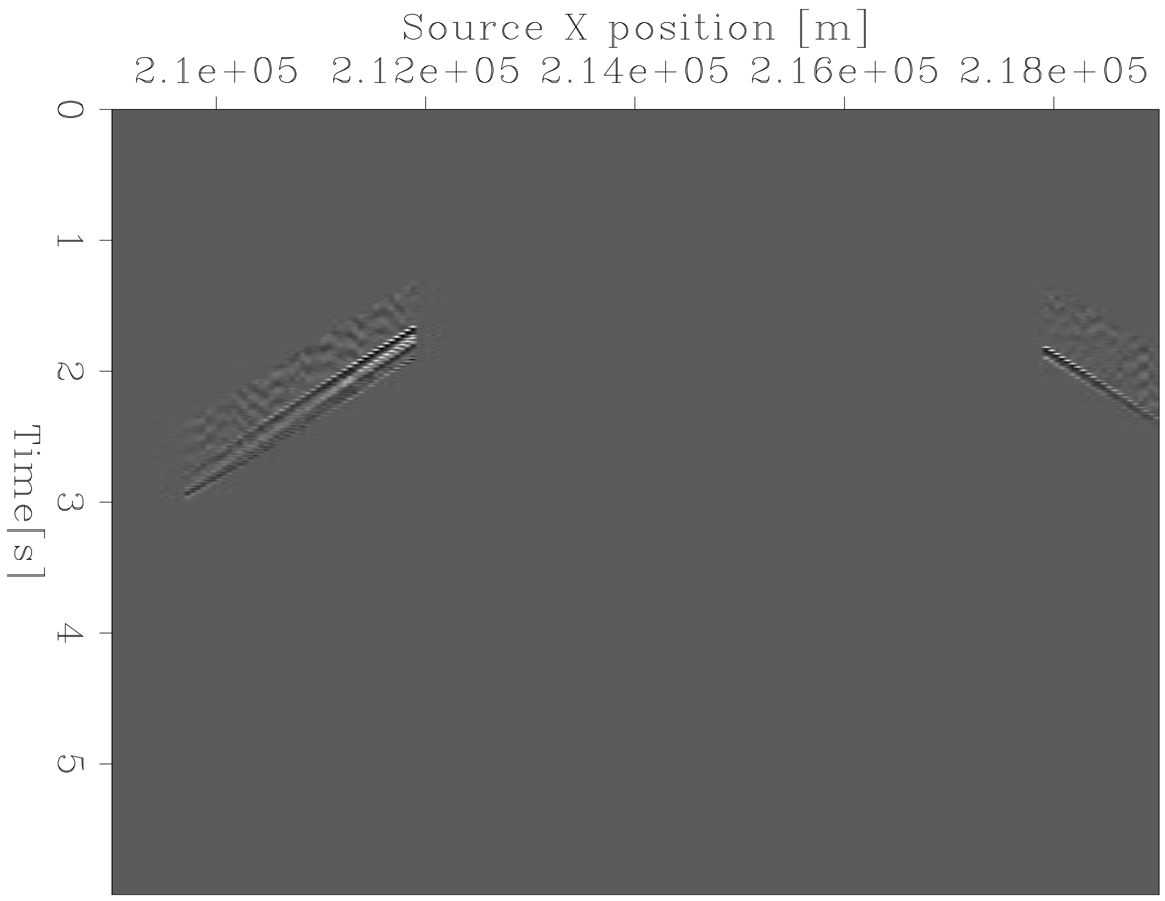


Figure B.4: Geophone data window (isolating the refraction events) used for estimating PZ-summation filter. [**CR**] $\boxed{\text{appendix2/. Zwindow}}$

that the ray parameter is constant in the window we minimize. We can then remove this effect from $\hat{a}(f)$ to get $a(f)$. We can estimate the ray parameter p in the rest of the data which allows us to reuse $a(f)$ to separate the upgoing (figure B.5) and downgoing (figure B.6) components using equations B.1 and B.2.

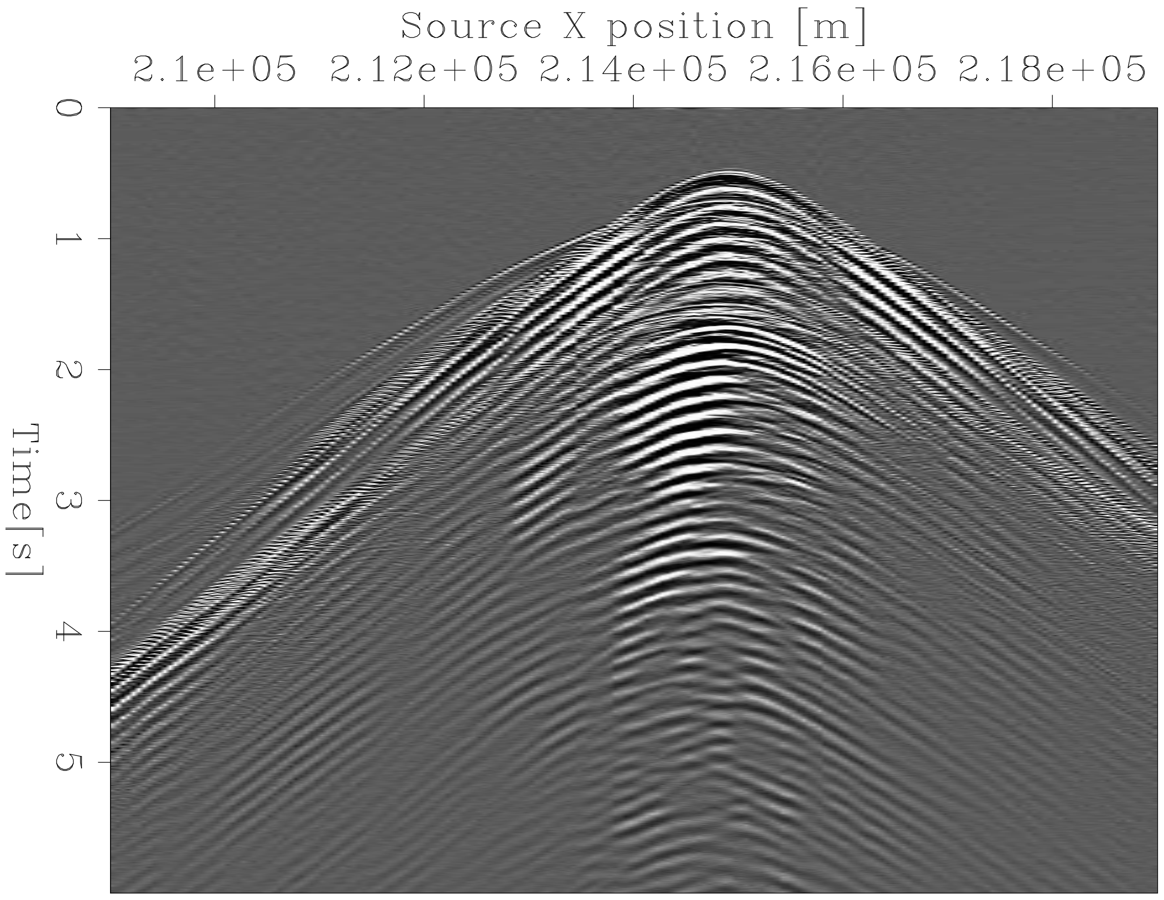


Figure B.5: Upgoing component output from PZ-summation. [**CR**] appendix2/. upgoing

Debubble and wavelet shaping

Picking the data wavelet

Once we have the downgoing component of the hydrophone data separated, we can estimate a source wavelet from a near-offset subset of it (figure B.7). First, we perform hyperbolic moveout (HMO) for each node gather to align the first arrival event across all offsets (figure B.8). We can then stack across all offsets to find a single wavelet

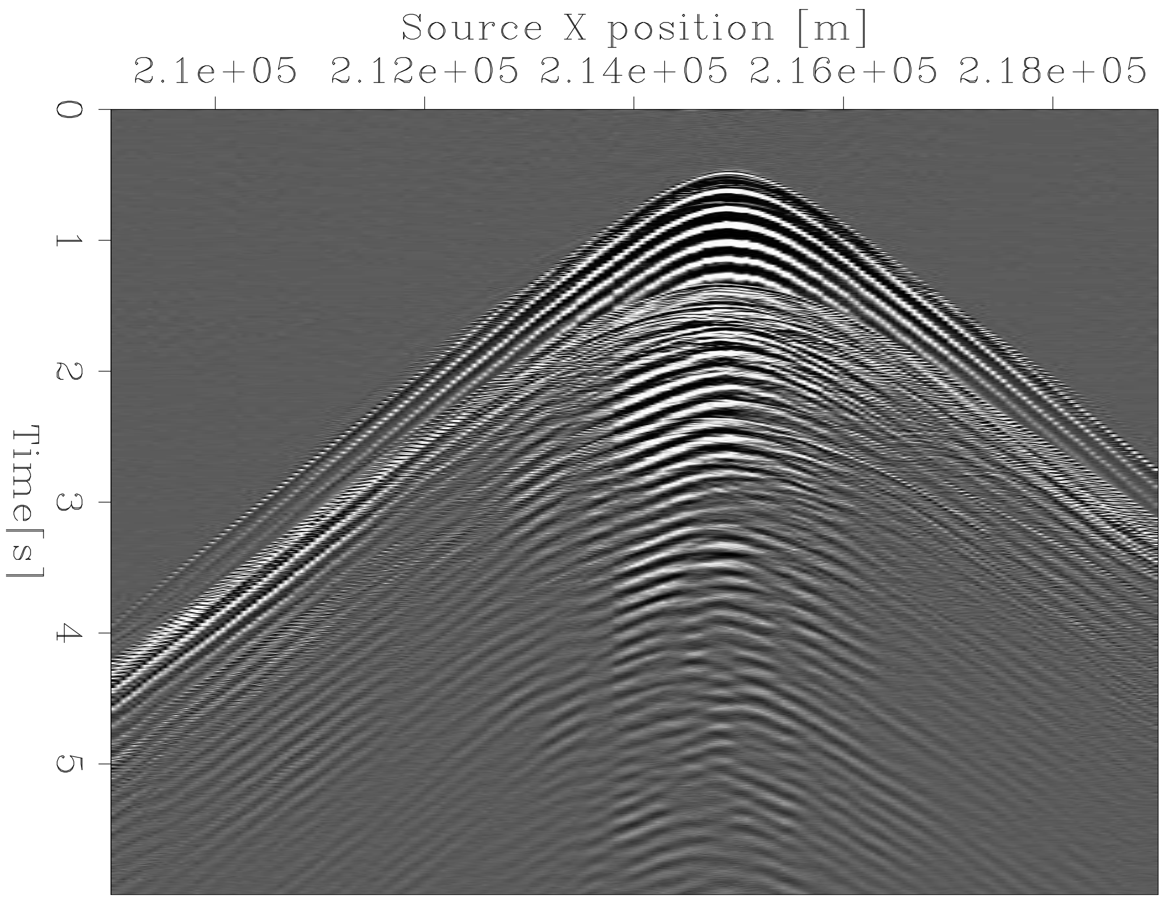


Figure B.6: Downgoing component output from PZ-summation. [**CR**] appendix2/. downgoing

representative of that node gather (figure B.10). To increase our averaging further, we can stack across all node gathers (figure B.11). To do this however, we need to align the first arrival event in each node gather to a common time position ($t_0 = 0$ for example). This means shifting based on the relative depth of each of the nodes respectively. We perform this shifting and stacking across nodes to find a final wavelet representative of the average of the sources actually used. In this wavelet we notice the bubble signature as periodic, fading pulses.

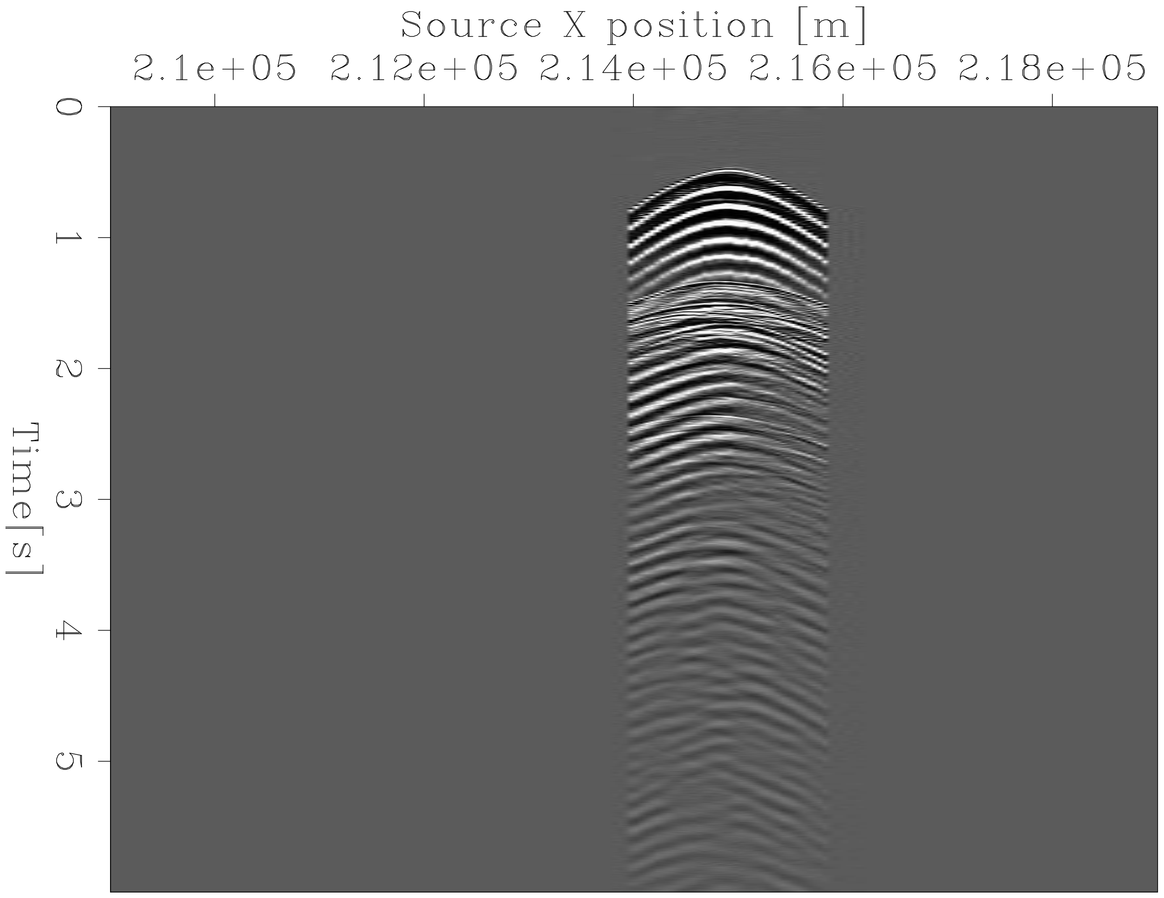


Figure B.7: Near offset (< 1000m) subset of figure B.6 chosen for estimating observed source wavelet. [**CR**] appendix2/. UncorrectedNear

Finding the shaping filter

In our synthetic data, we model using a simple 8[Hz] central frequency Ricker wavelet (see figure B.9). When we compare against the averaged observed data wavelet (figure B.11), we can see that there are some significant differences that warrant the use of a

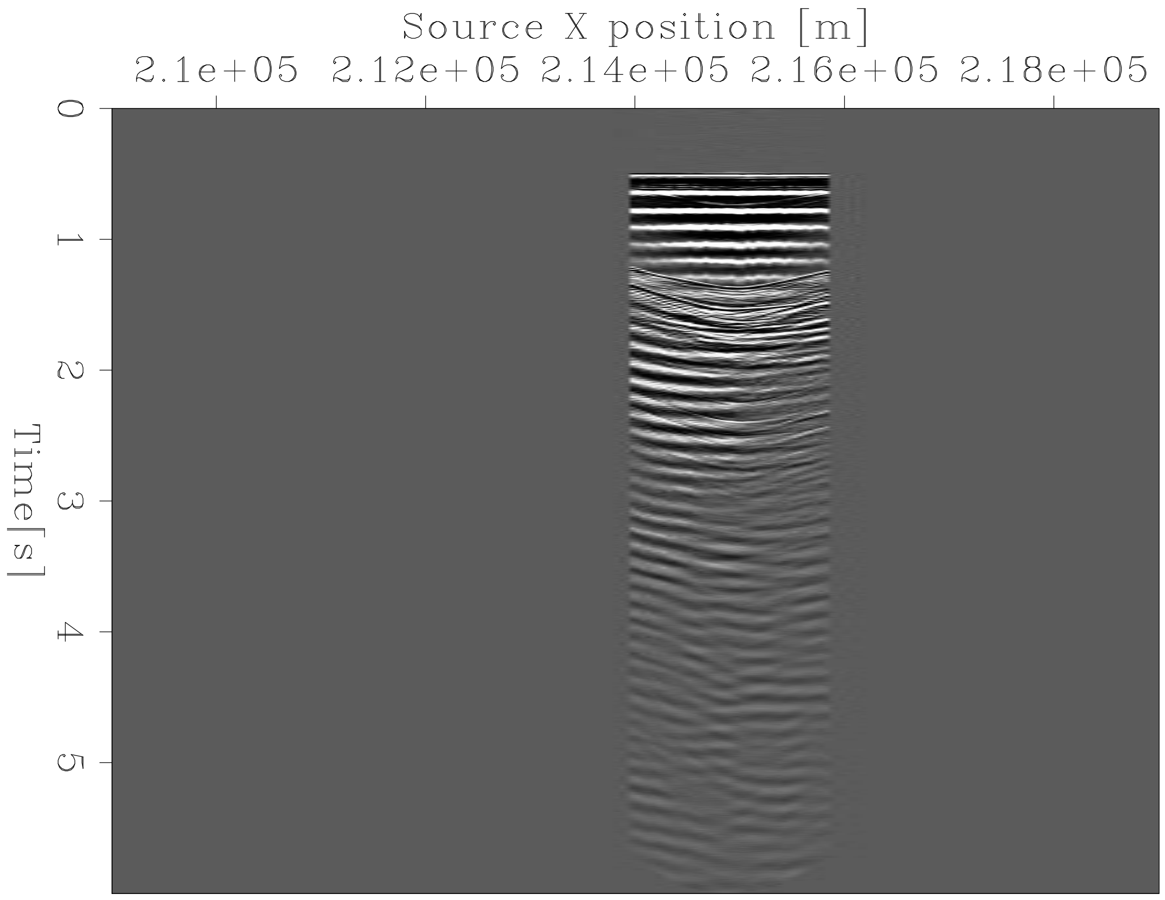


Figure B.8: 1500 m/s HMO correction applied to data in figure B.7. [**CR**] $\boxed{\text{appendix2/. HMOcorrected}}$

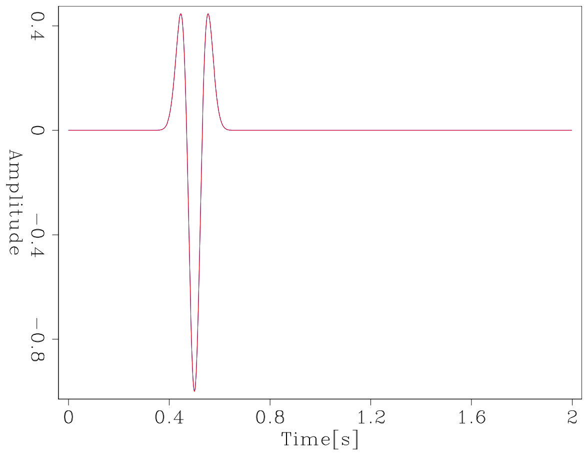


Figure B.9: Wavelet used in synthetic data modeling. [**CR**] $\boxed{\text{appendix2/. synWavelet}}$

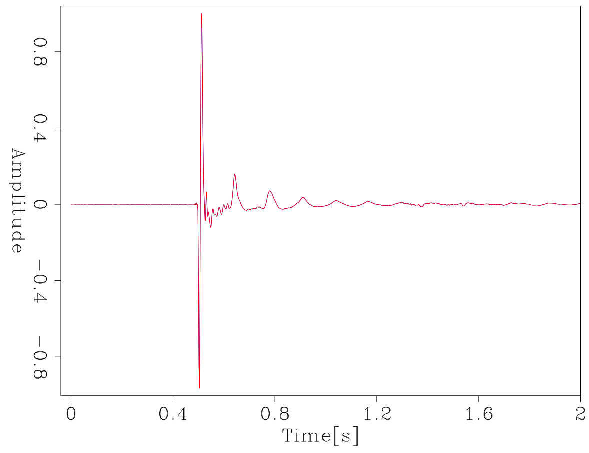


Figure B.10: Wavelet produced from stacking across traces in figure B.8. [**CR**] $\boxed{\text{appendix2/. Node3StackedWavelet}}$

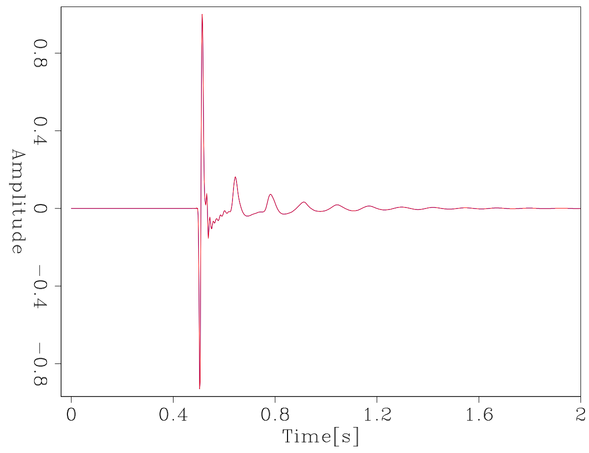


Figure B.11: Observed source wavelet built from average of wavelets extracted from 381 node gathers (just as in figure B.10). [**CR**] $\boxed{\text{appendix2/. AverageStackedWavelet}}$

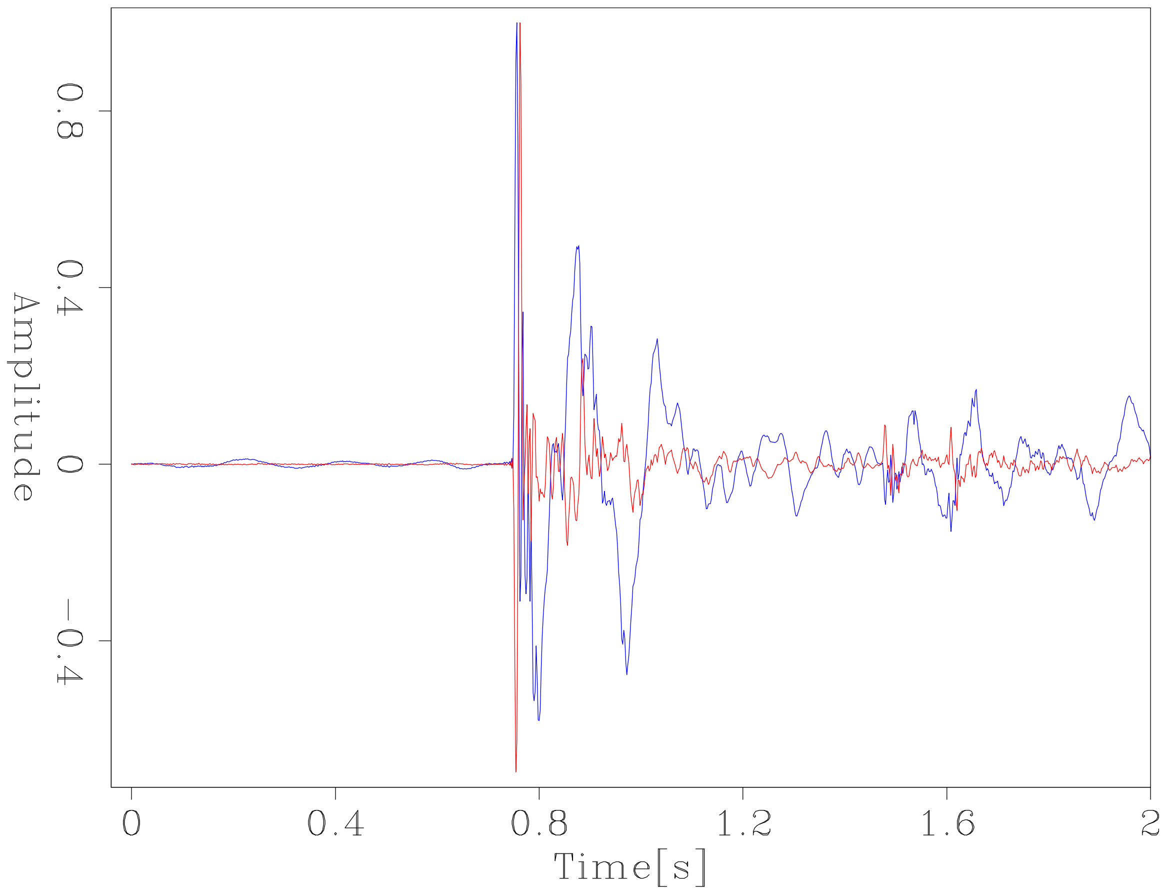


Figure B.12: Example trace before (red) and after (blue) shaping filter applied. [**CR**] appendix2/. ShapedTraceCompare

shaping filter. We estimate the filter that shapes the averaged observed data wavelet to the synthetic wavelet (see Yilmaz (1987)), and then apply to the full dataset (figures B.13 and B.14). When we compare figure B.6 with figure B.13, we can see that the bubble removal is effective, and that the frequency and phase content has become more similar to what we would expect given the lower frequency Ricker wavelet to which we match.

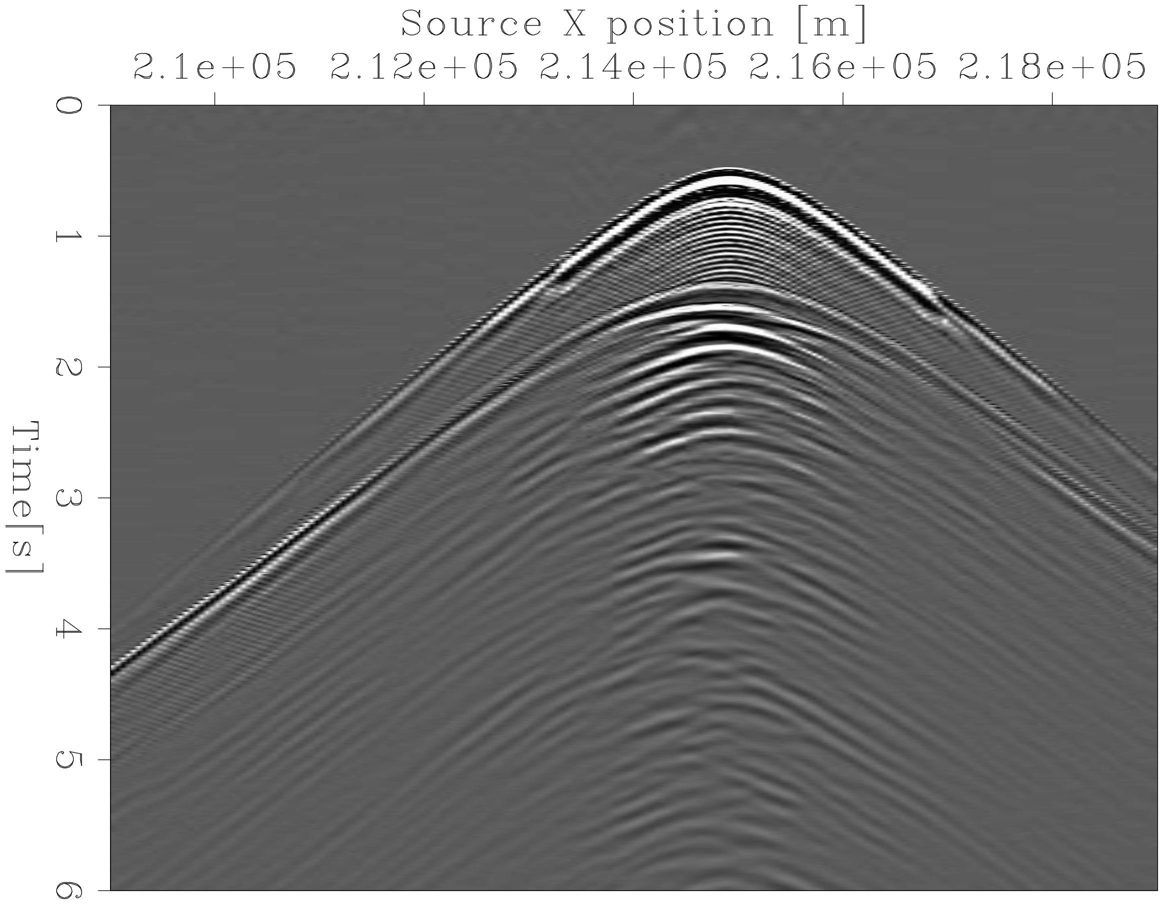


Figure B.13: Downgoing hydrophone data after shaping filter applied. [**CR**] appendix2/. ShapedDOWNData

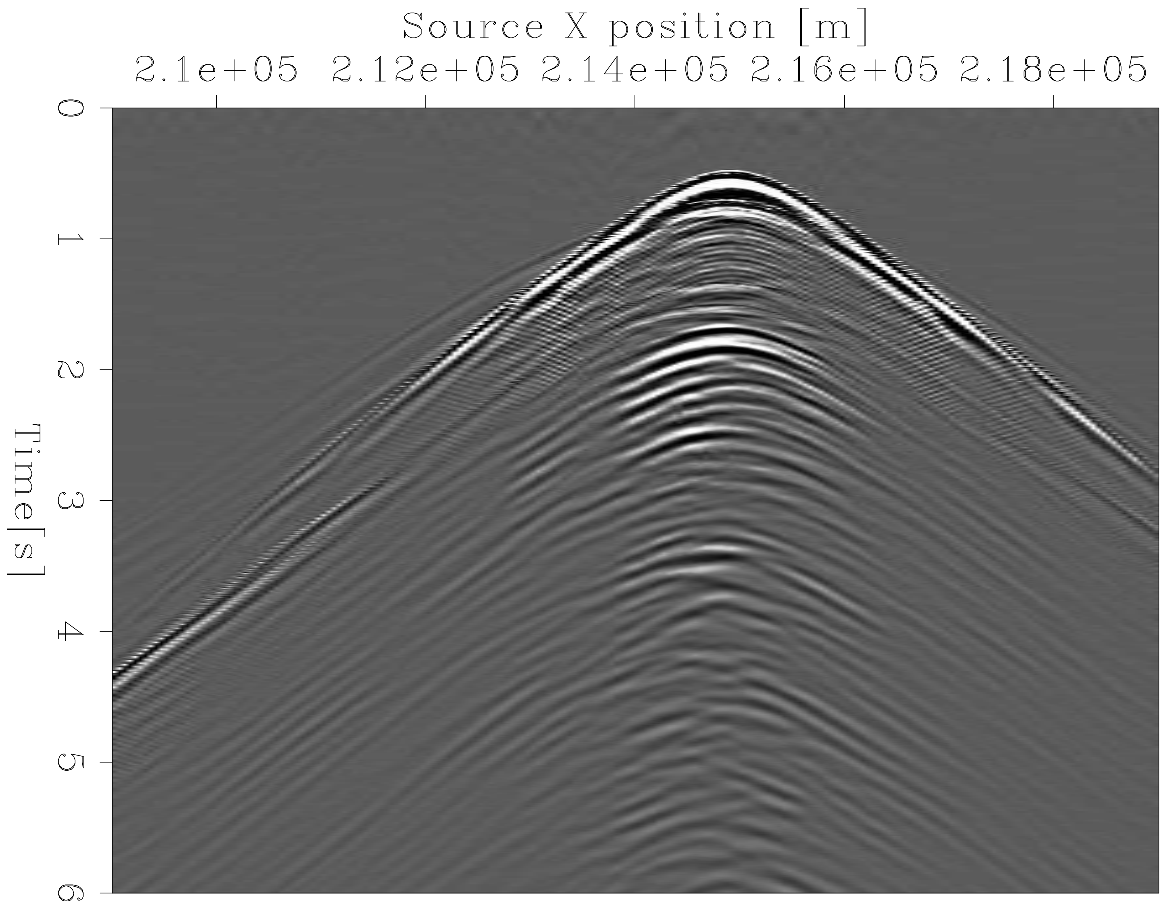


Figure B.14: Upgoing hydrophone data after shaping filter applied. [**CR**] appendix2/. ShapedUPData

Bibliography

Aster, R. C., B. Borchers, and C. H. Thurber, 2013, Chapter six - iterative methods, *in* Aster, R. C., B. Borchers, and C. H. Thurber, eds., Parameter Estimation and Inverse Problems (Second Edition), 141 – 168, Academic Press, second edition ed.

Barnier, G. and B. Biondi, 2015, Addressing the effects of inaccurate top-salt delineation on subsalt seismic imaging: SEP-158.

Biondi, B., E. Biondi, M. Maharramov, and Y. Ma, 2015, Dissection of the full-waveform inversion hessian: SEP-Report, **160**, 19–38.

Biondi, E. and S. A. Levin, 2014, Application of the up-down separation using PZ calibration filter based on critically refracted waves: SEP-Report, **155**, 119–132.

Bunks, C., F. M. Saleck, S. Zaleski, and G. Chavent, 1995, Multiscale seismic waveform inversion: GEOPHYSICS, **60**, 1457–1473.

Burger, M., 2003, A framework for the construction of level set methods for shape optimization and reconstruction: Interfaces and Free boundaries, **5**, 301–330.

Cox, B. E. and D. Verschuur, 2001, Data-driven tomographic inversion of focusing operators, *in* SEG Technical Program Expanded Abstracts 2001, 722–725, Society of Exploration Geophysicists.

Esser, E., L. Guasch, T. van Leeuwen, A. Y. Aravkin, and F. J. Herrmann, 2018, Total variation regularization strategies in full-waveform inversion: SIAM Journal on Imaging Sciences, **11**, 376–406.

Etgen, J. T., D. J. Foster, and Y. Zhang, 2016, Introduction to the special section: Subsalt imaging: The Leading Edge, **35**, 226–227.

Fichtner, A., 2010, Full seismic waveform modelling and inversion. Advances in Geophysical and Environmental Mechanics and Mathematics: Springer Berlin Heidelberg.

Grion, S., R. Exley, M. Manin, X. Miao, A. Pica, Y. Wang, P. Granger, and S. Ronen, 2007, Mirror imaging of obs data: first break, **25**, 37–42.

Guo, Z. and M. de Hoop, 2013, Shape optimization and level set method in full waveform inversion with 3d body reconstruction: SEG Technical Program Expanded Abstracts, 1079–1083.

Kadu, A., T. V. Leeuwen, and W. Mulder, 2016, A parametric level-set approach for seismic full-waveform inversion, 1146–1150.

———, 2017a, Parametric level-set full-waveform inversion in the presence of salt bodies, 1518–1522.

Kadu, A., T. van Leeuwen, and W. A. Mulder, 2017b, Salt Reconstruction in Full-Waveform Inversion With a Parametric Level-Set Method: IEEE Transactions on Computational Imaging, **3**, 305–315.

Knopoff, L. and A. F. Gangi, 1959, Seismic reciprocity: Geophysics, **24**, 681–691.

Larson, R., 2009, Elementary algebra. Available 2010 Titles Enhanced Web Assign Series: Cengage Learning.

Leveille, J. P., I. F. Jones, Z.-Z. Zhou, B. Wang, and F. Liu, 2011, Subsalt imaging for exploration, production, and development: A review: GEOPHYSICS, **76**, WB3–WB20.

Lewis, W., B. Starr, D. Vigh, et al., 2012, A level set approach to salt geometry inversion in full-waveform inversion: Presented at the 2012 SEG Annual Meeting.

Maciejewski, M., W. Surtel, and T. Malecka-Massalska, 2012, Level-set image processing methods in medical image segmentation: 2012 Joint Conference New Trends In Audio Video And Signal Processing: Algorithms, Architectures, Arrangements And Applications (NTAV/SPA), 39–41.

Maharramov, M., 2016, Time-lapse inverse theory: PhD thesis, Stanford University.

Maharramov, M. and S. A. Levin, 2015, Total-variation minimization with bound constraints: arXiv e-prints, arXiv:1505.05694.

Marquardt, D., 1963, An algorithm for least-squares estimation of nonlinear parameters: Journal of the Society for Industrial and Applied Mathematics, **11**, 431–441.

Melbo, A. H., J. O. Robertsson, and D.-J. van Manen, 2002, Pz calibration by applying the equation of motion to critically refracted waves, *in* SEG Technical Program Expanded Abstracts 2002, 1030–1033, Society of Exploration Geophysicists.

Osher, S. and J. A. Sethian, 1988, Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations: Journal of computational physics, **79**, 12–49.

Santosa, F., 1996, A level-set approach for inverse problems involving obstacles:

ESAIM: Control, Optimisation and Calculus of Variations, **1**, 1733.

Shen, X., 2010, Near-surface velocity estimation by weighted early-arrival waveform inversion, *in* SEG Technical Program Expanded Abstracts 2010, 1975–1979, Society of Exploration Geophysicists.

Shen, X., I. Ahmed, A. Brenders, J. Dellinger, J. Etgen, and S. Michell, 2017, Salt model building at atlantis with full-waveform inversion.

Tsai, A., A. Yezzi, W. Wells III, C. Tempany, D. Tucker, A. Fan, W. E. Grimson, and A. S. Willsky, 2003, A shape-based approach to the segmentation of medical imagery using level sets.

Virieux, J. and S. Operto, 2009, An overview of full-waveform inversion in exploration geophysics: GEOPHYSICS, **74**, WCC1–WCC26.

Xia, H., P. Tucker, and W. Dawes, 2010, Level sets for cfd in aerospace engineering: Progress in Aerospace Sciences, **46**, 274 – 283.

Yilmaz, O., 1987, Seismic data processing: Society of Exploration Geophysicists.