Copyright
by
Debanjan Datta
2017

The Dissertation Committee for Debanjan Datta
certifies that this is the approved version of the following dissertation:

Implementing efficient Global Optimization methods in Full Waveform Inversion

Committee:

Mrinal K. Sen, Supervisor

Kyle Spikes

Sergey Fomel

Steve Grand

Kirk Jordan

Implementing efficient Global Optimization methods in Full Waveform Inversion

by

Debanjan Datta

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2017

Dedicated to my family.

Acknowledgments

I owe this thesis to a lot of people without whom I would not be able to reach this far. The list definitively starts with my advisor Dr. Mrinal Kanti Sen. Starting from the time we met at National Geophysical Research Institute, he has been a constant source of inspiration, constructive criticism and enthusiasm. Our interactions helped me generate most of the ideas presented in this thesis and his constant support throughout the duration of my Ph.D. was instrumental in attaining the current state of quality for this document.

I owe a great deal of gratitude to my Committee members Dr. Mrinal Sen, Dr. Kyle Spikes, Dr. Stephen Grand, Dr. Sergey Fomel & Dr. Kirk Jordan. They did not just serve the role of my committee members but also passed on their expertise of different fields like Seismology, Rock Physics, Seismic Imaging, Seismic Processing and High Performance Computing. This thesis would not be possible without the strong foundations of the above-mentioned subjects imparted by my committee members through courses or discussions.

Words cannot express the amount of gratitude I owe different professors who taught me various courses and armed me with the knowledge needed to finish my Ph.D. Dr. Sen built the foundation stone for Inverse Theory, Anisotropy, and Seismic Imaging. Dr. Grand's course on Seismology II was in-

strumental in the understanding of whole earth processes and governing physical principles of the earth. Dr. Spikes built the perfect bridge between seismic and well log data with his courses on Rock Physics and Quantitative Seismic interpretation. Dr. Fomel's course not only provided the concepts and nuances of seismic processing but also introduced the excellent Madagascar package as a tool for reproducible research in the field. Both Dr. Spikes' and Dr. Fomels' assignments on field datasets prepared with Thomas Hess were instrumental in getting a first-hand experience of the corresponding methods learned in class. Dr. George Biros taught the subjects of Linear Algebra and Parallel Computing which form the building block of two chapters of this thesis. Dr. Charles Jackson's course on Uncertainty Estimation made me understand the importance of quantifying uncertainty in geophysical interpretations.

The list of acknowledgments would not be complete without mentioning my fellow graduate students who I have interacted with during my time as a Ph.D. student. The interactions led to countless refinements to my research and also kept me informed on cutting edge research in related fields. I am indebted to Dr. Qi Ren, Dr. Russel Carter, Dr. Meijuan Jiang, Dr. Zeyu Zhao, Dr. Han Liu, Dr. Junzhe Sun, Reetam Biswas, Anthony Barone, Janaki Vamaraju, Hala Akhtari, Badr Alulaiw, Alejandro Garza, Armando Calderon, Pan Bei, Elliot Dahl & Barry Borgman. They all have been some of the best minds I have had the pleasure to interact with.

This thesis would not be complete without mentioning the role of collaborators. I would like to thank Dr. Piyoosh Jaysaval & Dr. Adrien Arnulf

who introduced and mentored me to the Schur complement approach. Piyoosh's expertise on direct solvers proved invaluable when implementing the Schur complement method. Adrien's insightful edits to manuscripts made me confident in the quality of the manuscript. I would also like to thank Dr. Scott Morton and Dr. Faqi Liu with whom I collaborated on two of my chapters in this thesis. Scott and Faqi were extremely knowledgeable and their feedback on my research went a great deal to improve the methods mentioned in this thesis.

It would be a great injustice if I fail to thank the wonderful people I met at IBM Research during my summer internships. The most significant person on the list is Dr. Kirk Jordan. I started my internship project without any significant experience in High Performance Computing but with his continuous mentoring and guidance, I was able to not only able to pick key concepts and rethink a problem focusing on data rather than just algorithms. I would also like to thank David Appelhans, Constantinos Evangelinos and Vipin Sachdeva of IBM Research who I collaborated with on multiple projects in IBM Research.

A concerned effort from several non-teaching staff made sure my time at UT passed like a well oiled machine. Words cannot express the contribution of the graduate co-ordinator, Phillip Guerrero. He ensured that I finished all required formalities before even sitting down to write this thesis. Phillip's calm nature with good planning abilities make every graduate student go through a painless graduation process. I also thank Teresita Gonzalez of the Jackson

School for making sure my paperwork and payments were all up to date. At the Institute of Geophysics, I am indebted to the system administrator Mark Weiderspahn. He was the one who made sure all the servers were running smoothly and dependencies were satisfied. His constant emails about back up and system status made me plan our work well in advance. I also thank the administrative staff at the institute namely, Judy Samson, Rosalind Gamble, Anisa Abdulkader and Rikk Rodriguez for making sure any formalities and purchases were done on time.

Lastly, I thank my parents Debasis & Anindita Datta, without whom I would not have been what I am. Their constant support through video and phone calls did not allow me to miss home in the slightest. Thank you, mom and dad.

Implementing efficient Global Optimization methods in Full Waveform Inversion

Publication No. _____

Debanjan Datta, Ph.D.
The University of Texas at Austin, 2017

Supervisor: Mrinal K. Sen

Full Waveform Inversion (FWI) is slowly becoming the standard for velocity estimation from seismic data. It uses the full wave equation to estimate high resolution models capable of recovering finer details compared to traditional methods like tomography. FWI is constructed as a least squares optimization problem that iteratively updates an initial model until the misfit between modeled and observed data are within a pre-specified tolerance. The local nature of FWI makes it sensitive to the choice of starting model which has to be in the global minimum trough of the misfit function. If the starting model lies outside the global minimum trough, FWI converges to a local minima giving an incorrect estimation of subsurface properties. In this thesis, I eliminate the sensitivity to the starting model when performing FWI. This is done by re-casting FWI into a global optimization problem using a combination of sparse parameterization methods. The starting model is generated by a global optimization method that is not sensitive to the starting model.

The result from the global optimization method is then used as a starting model for conventional FWI using a dense set of parameters. In chapter 2, I apply global optimization with a sparse parameterization technique for 2D models using a combination of interfaces and velocities. Chapter 3 extends the sparse parameterization for salt bodies using ellipse sets and level-set methods. Lastly, chapter 4 extends sparse parameterization in three dimensions to perform FWI without a starting guess. I demonstrate that the proposed approaches are not sensitive to the source of starting model and discuss some sources of uncertainty.

Another issue that prevents implementation of FWI on a regular basis is its computational cost. Expensive forward evaluations need to be performed multiple times across several iterations to obtain the final model. It is of particular significance in 3D where in addition to posing a computational cost problem, it also creates a data reversal problem when computing the gradient. I propose two approaches to reduce the computational cost of forward modeling and gradient computation in the time and frequency domains. The frequency domain approach discussed in chapter 5 exploits the contribution of unchanging zones in an inversion workflow to be reused over subsequent iterations of FWI. The final chapter discusses how to run a gradient computation on GPUs using faster interconnect and on-node memory. I provide a template to apply it to a general adjoint problem with specific example on FWI. I obtain a 20-40% and 82-207% improvement in runtimes in the frequency and time domain, respectively, using the proposed methods.

Table of Contents

Acknowledgments **v**

Abstract **ix**

List of Tables **xvi**

List of Figures **xvii**

Chapter 1. Introduction **1**

1.1 Background Theory 3
1.2 Full Waveform Inversion 4
1.2.1 Forward Modeling 4
1.2.2 Cost Function 6
1.2.3 Gradient Computation 6
1.2.4 Optimization 7
1.3 Limitations 8
1.3.1 Cycle Skipping 8
1.3.2 Computational issues 10
1.4 My Contributions 10
1.5 Previous Work 11
1.6 Outline 11

Part I Global Optimization in FWI **13**

Chapter 2. Estimating a starting model for full waveform inversion using a global optimization method **14**

2.1 Summary 14
2.2 Introduction 15

2.3 Theory 20
2.3.1 Estimation of a starting model using VFSA 21
2.3.2 Very fast simulated annealing 25
2.3.3 Gradient FWI from VFSA model 27
2.4 Results 29
2.4.1 Field Data I: Krishna Godavari Basin, India 35
2.4.2 Field Data II: Mahanadi Basin, India 39
2.5 Discussion 46
2.5.1 Sensitivity towards the starting model and effect of priors 46
2.5.2 Choice of VFSA parameters 50
2.5.3 Computational overhead from VFSA inversion 51
2.5.4 Extension to 3D 53
2.5.5 Multi-VFSA for uncertainty estimation 54
2.5.6 Other global optimization methods 55
2.6 Conclusions 55

Chapter 3. Salt shape estimation using shape parameterization and Full Waveform Inversion 57

3.1 Summary 57
3.2 Introduction 58
3.3 Theory 62
3.3.1 Full Waveform Inversion 62
3.3.2 Very Fast Simulated Annealing 62
3.3.3 Sparse Parameterization of sediments: Velocity Interface Approach 64
3.3.4 Sparse parameterization of salt bodies using ellipses . . 64
3.3.5 FWI using Adjoint state gradient 66
3.4 Numerical Results 69
3.4.1 SEG-EAGE salt model 70
3.4.2 Sigsbee 2A model 81
3.4.3 Modified SEG-SEAM Phase I model 87
3.5 Discussions 93
3.5.1 Uncertainty from different starting models 93

3.5.2 Uncertainty from inaccurate salt velocities 96
3.5.3 Uncertainty from Concave Hull 96
3.5.4 Uncertainty from the chosen number of ellipse 96
3.6 Conclusions 97

Chapter 4. Global 3D Full Waveform Inversion with sparse parameterization 98
4.1 Summary 98
4.2 Introduction 99
4.3 Theory 100
4.4 Results 102
4.4.1 Validation test on toy model 102
4.4.2 SEG EAGE Overthrust Model 102
4.5 Conclusions 105

Part II Computational Issues in FWI 111

Chapter 5. Fast 2D Full Waveform Inversion in the frequency domain using Schur Complement Approach 112
5.1 Summary 112
5.2 Introduction 113
5.3 Theory 117
5.4 Results 124
5.4.1 FWI for a simple synthetic model 126
5.4.2 Sigsbee2A 127
5.5 Discussions 134
5.5.1 Extension to elastic and anisotropic cases 134
5.5.2 3D extension 134
5.5.3 Undulating seafloor 134
5.6 Conclusions 135

Chapter 6. Efficient Gradient and Image Computation using a two level checkpointing method for adjoint problems using NVLINK connected CPU-GPUs 136
6.1 Summary . . . 136
6.2 Introduction . . . 137
6.2.1 Overview . . . 137
6.2.2 Full Waveform Inversion . . . 140
6.2.3 Related Work . . . 142
6.3 Theory . . . 145
6.3.1 Two Level Checkpointing . . . 145
6.3.2 Compute model . . . 147
6.4 Asynchronous Implementation . . . 150
6.4.1 Forward and re-population stages . . . 150
6.4.2 Adjoint and reconstruction stages . . . 152
6.4.3 Strategies for selecting optimal checkpointing parameters 156
6.4.4 Memory Constraints . . . 157
6.4.5 Performance model when partial data transfer is hidden 158
6.4.6 Benefits of faster interconnects and high speed on-node memory . . . 162
6.5 Results . . . 163
6.5.1 Code set up . . . 164
6.5.2 Optimal checkpointing values . . . 165
6.5.3 Runtimes (Predicted vs Computed) . . . 166
6.5.4 Runtimes for different Configuration . . . 167
6.5.5 Scaling . . . 168
6.6 Conclusions . . . 169

Chapter 7. Summary 172
7.1 Future Work . . . 175

Appendices 177

Appendix A. Swarm optimization in FWI **178**
A.0.1 Particle Swarm Optimization 178
A.0.2 PSO gradient hybrid optimization 179
A.1 Results 180
A.2 Discussions and conclusions 182
A.2.1 PSO tuning factors 182
A.2.2 Population Size and iterations 182
A.2.3 Quality of the starting model 183
A.3 Conclusions 183

Bibliography **185**

Vita **205**

List of Tables

4.1 Table showing VFSA results for a 4-layered toy model with 3 interfaces. The true values are shown in square brackets and the search space is given in parentheses 103

5.1 Runtimes and memory requirements per frequency for forward modeling using the standard FDFD approach. 132

5.2 Runtimes and memory requirements per frequency for forward modeling using the Schur Complement approach 133

List of Figures

1.1 Flowchart showing the Full Waveform Inversion algorithm . . 5

1.2 Schematic of cycle-skipping artifacts in FWI after Virieux and Operto (2009). The solid black line represents a monochromatic seismogram of period T as a function of time. The upper dashed line represents the modeled monochromatic seismograms with a time delay greater than T/2. In this case, FWI will update the model such that the $n+1th$ cycle of the modeled seismograms will match the nth cycle of the observed seismogram, leading to an erroneous model. In the bottom example, FWI will update the model such that the modeled and recorded nth cycle are in-phase because the time delay is less than T/2. 9

2.1 A schematic showing the phenomenon of cycle skipping in FWI. (a) shows an observed trace one wishes to match. (b) and (c) show two trial traces shifted by different time samples compared to (a). Using (b) and (c) as limits on either side, a misfit vs shift curve is shown in (d) where the presence of several local minima troughs can be seen. These local minima troughs can be avoided if the observed and trial traces are in agreement within half the propagating wavelength of the wave. If not, the convergence happens in a local minimum. 18

2.2 Figure showing the performance of cubic (Blue) and non oscillatory splines (Red): The black dots are the nodes used for interpolation and the blue and red curves show the result of spline interpolation and non oscillatory interpolation respectively. Cubic splines show considerable oscillations which are not present in non-oscillatory splines. This allows representation of an interface more accurately using fewer points. 23

2.3 An example velocity model derived from the proposed parameterization. I build the velocity model using five interfaces. The nodes in each interface are shown with white circles while the dotted black line show the interface after interpolation. The velocities v_{up}^k and v_{dn}^k are fixed across the interfaces and the velocities between interfaces are estimated using linear interpolation. 24

2.4 A flowchart of the proposed sparse parametrization method using non oscillatory splines for VFSA inversion to obtain a good starting model. 25

2.5 (a) The true SEG-EAGE Overthrust model. (b) A strategy for fixing the search space of VFSA parameters. I generate a zero offset section of the SEG-EAGE Overthrust model and pick 4 interfaces. I pick nodes for interfaces in time following seismic events. The picked nodes are shown by white circles and the events are shown by dotted white lines. 31

2.6 (a) Figure showing a random starting model for VFSA. The interfaces in the starting model are completely random and do not show any resemblance to the final model after VFSA shown in (b). (b)The velocity model obtained after 200 iterations of VFSA using 4 picked interfaces from Figure 2.5b. The estimated velocity model is fairly smooth with a few marked interfaces. (c) The velocity model obtained after 80 iterations of gradient-based FWI using the model from a smoothed version of (b) as the starting model. This velocity model is able to retrieve all high wavenumber features of the velocity model and is very similar to the true model shown in Figure 2.5a. 32

2.7 Comparison of synthetic seismograms for a single shot at x=100m for the SEG-EAGE Overthrust model. (a) shows the seismogram generated from the model generated by VFSA. (b) shows the seismogram obtained from the final model after FWI and (c) shows the seismogram from the true model. Trace comparison of a few offsets are shown in Figure 2.8 33

2.8 Comparison of traces between the seismograms (at 10Hz) computed from the true model(red), VFSA model (blue) and FWI model (black) at different offsets in Figure 2.7. (a) compares the seismogram responses at an offset of 2.5km while (b) shows the same at 7.5km offset and (c) at 10km. In (a), (b) and (c) one can observe that the VFSA seismogram is within half the dominant period of the true model response. There are a few mismatched events that are recovered after gradient FWI. . . 34

2.9 I pick the nodes for the interfaces from a near offset section of KG data. I selected nodes for three interfaces shown in black circles and the initial interpolated interface is shown by a dotted yellow lines. 36

2.10 A random starting velocity model for KG data using 3 picked interfaces from Figure 2.9. 36

2.11 The velocity model obtained for KG data after 200 iterations of VFSA using 3 picked interfaces from Figure 2.9. The estimated velocity model after VFSA contains just two interfaces. This velocity model is smoothed and used as the starting model for a gradient-based FWI. 38

2.12 The velocity model obtained after 80 iterations of gradient-based FWI. This velocity model is able to delineate the BSR at the correct depth shown by black arrows. 39

2.13 Convergence curves for the KG dataset: (a) Normalized error vs iteration for VFSA iterations to obtain a sparse starting model for FWI. (b) Normalized error vs iteration for 80 iterations of FWI with the starting model from VFSA based minimization. 40

2.14 (a) shows a shot record from the KG dataset at x=2km while (b) shows the seismic response for the FWI model at the same location. They show very good agreement. (c) shows the comparison between the well log (blue) the FWI model (red) the VFSA model (black) and the smoothed VFSA model (green). 41

2.15 I pick the nodes for the interfaces from a near offset section of Mahanadi data where selected nodes for five interfaces shown in white circles and the events leading to the picks have been shown by dotted white lines. 43

2.16 A random starting velocity model for Mahanadi data using 5 picked interfaces from Figure 2.15. 44

2.17 The velocity model obtained for Mahanadi data after 200 iterations of VFSA using 5 picked interfaces from Figure 2.15. The estimated velocity model contains the optimal location of interfaces. This velocity model, after smoothing, is used as the starting model for a gradient-based FWI. 44

2.18 The velocity model obtained after 80 iterations of gradient-based FWI using the model from Figure 2.17 as the starting model. The seafloor is shown by the dotted black line and the model contains all high wavenumber features not contained in the VFSA model. 45

2.19 An overlaid image of the velocity model obtained after FWI for the Mahanadi data over its corresponding RTM image. The events in the RTM section are well correlated with velocity changes. 45

2.20 Convergence curves for the Mahanadi dataset: (a) Normalized error vs iteration for VFSA iterations to obtain a sparse starting model for FWI. (b) Normalized error vs iteration for 80 iterations of FWI with the starting model from VFSA based minimization. 47

2.21 Comparison of Common Image Gathers(CIGs) for the Mahanadi data (a) shows the CIGs from the VFSA model and (b) shows the CIGs from the model after FWI. CIGs from the VFSA model (a) are markedly flatter than the VFSA model (a) . . . 48

2.22 Comparison of the vertical wavenumbers contained the final model after FWI (blue) and the model after VFSA (red). One can see the gradient FWI (blue) filling up the missing wavenumbers from the VFSA model (red). This hints at the fact that the VFSA builds the low wavenumber component of the model and gradient FWI fills up mid to high wavenumbers thereby reconstructing an accurate velocity model. 49

2.23 Figure showing the stability of the inverted model to different initial guesses. I show 2 two interfaces from the synthetic test for clarity where the search window for each node is given by red rectangles and run 10 realizations of VFSA using different starting models. The starting guesses are shown in blue dots while the final interfaces obtained after each realization is shown with dotted black lines. The inverted interface appears to be invariant to the search space converging to nearly the same points with different starting guesses. The 4 velocities obtained in this inversion are also withing 5 percent of each other. 50

2.24 Convergence behavior using different decay factors. The red, blue and green curves show the convergence behavior using decay factors of 0.99, 0.98 and 0.80, respectively. Using a steep decay factor of 0.8 (green) leads to premature convergence to a non-optimal model. Using a slow cooling schedule of 0.99 (red) gives an optimal model but takes many more iterations to converge. The optimal cooling schedule of 0.98 (blue) converges to the same model as (red) in fewer iterations making it my choice of cooling schedule. 52

3.1 An ellipse-set representation of the SEG-EAGE salt model. The Ellipses in this example are barely touching each other leading to ragged edges which are not representative of a salt body in a sparse domain. 66

3.2 The ellipse set representation of SEG-EAGE salt model after taking the restricted concave hull of the ellipse sets. The edges are much smoother and better represent the true salt body. . 67

3.3 The true SEG-EAGE velocity model. The model spans 1349 points in the horizontal direction and 419 points in the vertical direction with a 15m spacing. 70

3.4 The estimated model obtained after 200 iterations of VFSA on the SEG-EAGE salt model. The model is parametrized by 4 interfaces and 3 Ellipses. The salt shape is very close to the true model, and the sediment velocity follows the low frequency trend of the sediments 72

3.5 Figure visualizing the search space of the salt body during the VFSA run for the SEG-EAGE salt model. The red zone is the cumulative representation of the salt body from all sampled models in VFSA 73

3.6 Relative error vs iteration during VFSA for the SEG-EAGE salt model. The initial iterations have oscillating error as worse models are being accepted, but the later iterations refine the model by only accepting models with a lower misfit. The acceptance to forward evaluation ratio was about 14 percent . . 74

3.7 Comparison between the salt shape of the True SEG-EAGE vs VFSA estimated salt. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The VFSA estimated salt body is fairly close to the true salt body. . . . 75

3.8 Comparison of seismogram with a central frequency of 3 Hz from the true SEG-EAGE model on the left and VFSA estimated model on the right. The seismograms appear to be free from cycle skipping. 76

3.9 Comparison of traces at trace no 400 with the true trace shown in blue and the VFSA trace shown in red. The seismograms are pretty closely matched and show no evidence of cycle skipping. 77

3.10 The final estimated velocity from SEG-EAGE model after 60 iterations of partitioned Level set FWI using the model from Figure 3.4 as the starting model. The salt body is better estimated after FWI and the sediments are also recovered fairly well. 78

3.11 Comparison between the salt shape of the True SEG-EAGE vs FWI model. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The FWI salt body shows marked improvement over the VFSA estimate. 79

3.12 Comparison between the vertical velocity profile between the true SEG-EAGE vs FWI model. 80

3.13 The true Sigsbee2A velocity model. The model spans 1065 points in the horizontal direction and 401 points in the vertical direction with a 20m spacing. 82

3.14 Figure visualizing the search space of the salt body during the VFSA run for the Sigsbee2A salt model. The red zone is the cumulative representation of the salt body from all sampled models in VFSA. 83

3.15 The estimated model obtained after 200 iterations of VFSA for the Sigsbee2A model. The model is parametrized by 4 interfaces and 5 Ellipses. The salt shape is very close to the true model and the sediment velocity follows the low frequency trend of the sediments. 83

3.16 Relative error vs iteration curve in the VFSA run for the Sigsbee2A model. The initial iterations have oscillating error as worse models are being accepted but the later iterations refine the model by only accepting models with a lower misfit. The acceptance to forward evaluation ratio was about 16 percent. . 84

3.17 Comparison between the salt shape of the true Sigsbee2A vs VFSA estimated salt. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The VFSA estimated salt body is fairly close to the true salt body. . . . 84

3.18 Comparison of seismogram with a central frequency of 3 Hz from the true Sigsbee2A model on the left and VFSA estimated model on the right. The seismograms appear to be free from cycle skipping. 85

3.19 Comparison of traces at trace no 800 with the true trace shown in blue and the VFSA trace shown in red. The seismograms closely match and show no evidence of cycle skipping. 85

3.20 The final estimated velocity from Sigsbee2A model after 60 iterations of partitioned level-set FWI using the model from Figure 3.15 as the starting model. The salt body is better estimated after FWI, and the sediments are also recovered fairly well. . . 86

3.21 Comparison between the salt shape of the True Sigsbee2A vs FWI model. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The FWI salt body shows marked improvement over the VFSA estimate. 86

3.22 Comparison between the vertical velocity profile between the true Sigsbee2A vs FWI model. 87

3.23 The modified SEG-SEAM velocity model. The model spans 1488 points in the horizontal direction ans 848 points in the vertical direction with a 10m spacing. 88

3.24 A rough shape of the salt body (v $\geq$ 3700) picked from a heavily smoothed true model. The picked shape can be represented by 9 ellipses shown by green dots. 89

3.25 Figure visualizing the search space of the salt body during the VFSA run for the SEG-SEAM salt model. The red zone is the total representation of the salt body from all sampled models in VFSA. 89

3.26 The estimated model obtained after 200 iterations of VFSA for the SEG-SEAM model. The model is parametrized by 4 interfaces and 9 Ellipses. The salt shape is very close to the true model and the sediment velocity follows the low frequency trend of the sediments. 90

3.27 Relative error vs iteration curve in the VFSA run for the SEG-SEAM model. The initial iterations have oscillating error functions as worse models are being accepted but the later iterations refine the model by only accepting models with a lower misfit. The acceptance to forward evaluation ratio was about 6 percent. 90

3.28 Comparison of seismogram with a central frequency of 3 Hz from the true SEG-SEAM model on the left and VFSA estimated model on the right. The seismograms appear to be free from cycle skipping. 91

3.29 Comparison between the salt shape of the True SEG-SEAM vs VFSA estimated salt. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The VFSA estimated salt body is fairly close to the true salt body. . . . 91

3.30 The final estimated velocity from SEG-SEAM model after 60 iterations of partitioned Level set FWI using the model from Figure 3.26 as the starting model. The salt body is better estimated after FWI and the sediments are also recovered fairly well. 92

3.31 Comparison between the salt shape of the True SEG-SEAM vs FWI model. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The FWI salt body shows marked improvement over the VFSA estimate. 93

3.32 The red part shows the part of the salt not covered by all five realizations of VFSA run for the SEG-EAGE model. The uncertainty is very small. 94

3.33 The red part shows the part of the salt not covered by all five realizations of VFSA run for the Sigsbee2A model. The uncertainty is pretty minimal compared to the salt body. 95

4.1 The 4 layer TOY model 105
4.2 (a) True 2D SEG-EAGE Overthrust Model (b) Inverted model after 100 iterations of VFSA 106
4.3 Error vs iterations for the SEG EAGE Overthrust Model . . . 107
4.4 (a) Modeled Seismogram from the true SEG-EAGE overthrust model and (b) Modeled seismogram from the VFSA result . . 108
4.5 (a) The True 2D slice from the SEG-EAGE Overthrust Model at $y = 3.5km$ (b) Starting slice from VFSA and (c) Inverted 2D model from (b) at four frequencies with 30 iterations per frequency of gradient FWI 109
4.6 (a) The True 2D slice from the SEG-EAGE Overthrust Model at $y = 5.7km$ (b) Starting slice from VFSA and (c) Inverted 2D model from (b) at four frequencies with 30 iterations per frequency of gradient FWI. 110

5.1 A typical Mass matrix showing banded structure. The matrix is square in shape with each dimension being N and the blue dots are the non zero elements while the white background represents zeros. 119
5.2 A prototype model being partitioned into background zone b which in this case is the water column. The background is not updated during inversion. The anomaly or sub-sea sediments are represented by a which is updated during inversion. 120
5.3 The partitioned matrix $\mathbf{M}$ being divided into 4 parts. The four parts represent the contribution of the background and anomalous zones to the propagating wavefield. 121
5.4 The two-layered model with a low velocity block in the middle. The length of the sea floor is variable ranging from 1 to 6 km.The model is discretized at 10m uniformly in vertical and horizontal directions. The horizontal span of the model is 20km while the vertical span ranges from 6km to 11km. 125
5.5 Seismic Response using standard FDFD of an identical 5 km-thick crustal model overlaid by varying water layer thicknesses W_d (a)W_d=6km (b)W_d=4km (c) W_d=2km and (d) W_d=1km. . 127
5.6 Seismic Response using Schur complement approach of an identical 5 km-thick crustal model overlaid by various water layer thicknesses W_d (a)W_d=6km (b)W_d=4km (c) W_d=2km and (d) W_d=1km. 128

5.7 (a) True synthetic model for FWI, (b) Starting model omits the anomalous block from the true model, (c)Inverted model after 3 groups of multi-scale FWI using standard FDFD approach & (d) Inverted model after 3 groups of multi-scale FWI using the schur complement approach. 129

5.8 Comparison of run-times betwen standard FDFD and Schur Complement approach 130

5.9 (a) True Sigsbee2A model for FWI (b) Starting model is the Sigsbee2A migration velocity (c) Inverted model after 3 groups of multi-scale FWI using standard FDFD approach & (d) Inverted model after 3 groups of multi-scale FWI using the schur complement approach. 132

5.10 Comparison of run-times between standard FDFD and Schur Complement approach. 133

6.1 Forward propagating states (red cubes) and the backward propagating adjoint steps (blue cubes) in a FWI problem. For each point in time, the forward and backward states need to be cross-correlated as shown by the * sign. Since the forward states are generated in reverse order of their use, one needs to save a subset of the forward states from which one can regenerate all forward states when needed for cross-correlation. 143

6.2 The vertical line represents the time scale from 0 to n_t in a one-level checkpointing scheme. Checkpoints are shown in blue horizontal lines while black horizontal lines show the intermediate steps which are not saved. During the forward pass all checkpoints are stored in a storage device. 144

6.3 The vertical line represents the time scale from 0 to n_t in a two-level checkpointing scheme. Thin black horizontal lines show the states whuch are not checkpointed. Coarse checkpoints are shown in red horizontal lines while fine checkpoints are shown in blue lines. All coarse checkpoints and fine checkpoints between 2 consecutive coarse checkpoints need to be stored. 146

6.4 Figure showing time reversal using a two-level checkpointing approach with $n_f = 5$ and $n_c = 10$ for a second-order in time adjoint scheme. The first foward sweep stores the coarse checkpoints and the fine checkpoints after the last coarse checkpoint while the next two reconstruction passes reconstruct the intermediate states shown in gray squares. The next pass repopulates the fine checkpoints before the last coarse checkpoint and the final 2 passes reconstruct from each of the fine checkpoints. Imagine many more time steps and only being able to fit the gray states in GPU memory, the blue states in system RAM, and the red states in a slower storage area. 149

6.5 Data movement during the forward and re-population phases of the two-level checkpointing for a second order in time adjoint problem with $n_f = 3$ across 4 time-steps. Red rectangles are the input states while blue is the output state. The dotted rectangle encompasses the buffers currently being transferred to the System RAM and the arrows denote the direction of data movement. The buffers in RAM are shown in white and green refers how much data has been transmitted. From $time = t$ to $time = t + 2$, the two buffers are being transferred into system RAM. Once the transfer is done, the same buffer is written into NVMe until the next coarse checkpoint overwrites it. The subsequent finer checkpoints are stored in other RAM buffers. 154

6.6 Data Movement during the reconstruction and adjoint phases with $n_f = 3$ for a second order in time adjoint problem. The two adjoint buffers are shown in grey and the circular arrows represents each buffer storing every alternate time-steps Forward and Reconstruction phases of the two-level checkpointing based FWI engine with $n_f = 3$. Red rectangles are the input states while blue is the output states. During evaluation of n_f time steps of adjoint 2 buffers are transferred to the GPU buffers enclosed by the thin black dotted rectangle. The adjoint states are computed with the remaining n_f buffers where forward states are already pre-computed. Thick dotted rectangle shows the collection of Forward and adjoint states at the same times. Similar to the forward propagation, data is read from NVMe while n_c kernel evaluations take place asynchronously. . 155

6.7 Figure showing the usable domain of streaming adjoint problem with different interconnects (NVlink vs PCIe) and storage (HDD vs NVMe) . The two axes show the size of a single checkpoint relative to available memory in the GPU and CPU. From memory constraints inequalities, I conclude that a faster interconnect will have a lower memory requirement in the CPU as well as RAM. The slower interconnects need more memory to compute in a streamlined function otherwise they end up waiting for data movement. 164

6.8 Predicted vs observed runtimes for a FWI gradient problem for different values of n_f using NVLink and NVMe. The predicted values were computed using equation 6.27 and appear to be in good agreement with observed values 166

6.9 Runtimes for a FWI engine for different storage and interconnects. NVLink and NVMe appear to have the best performance while HDD and PCIe are the worst. 168

6.10 Weak Scaling characteristics of the two-level checkpointing approach running with NVLink and NVMe for different values of n_f. 169
6.11 Strong Scaling characteristics of the two-level checkpointing approach running with NVLink and NVMe for different values of n_f. The curves suggest that as the work per processor decreases, the MPI overhead to exchange halos is greater than the time to compute the interior domain resulting in imperfect strong scaling. 170
6.12 Speed-up characteristic for different values of n_f. The speedups are consistent across different values of n_f. Large models may show better speed-up. 170

A.1 A random starting model generated from the picked interfaces using the sparse parameterization technique after Datta and Sen (2016) 181
A.2 The final inverted best solution for the Overthrust model after 10 iterations of hybrid PSO-FWI 182
A.3 The error vs forward evaluations plot for the best model for the Overthrust model. The blue lines show the gradient steps while the red curves show the PSO updates 183

Chapter 1

Introduction

The behavior of the propagating seismic wavefield inside the earth is a function of the source and the elastic properties of the earth. In a typical seismic survey, the wavefield can be sampled only at the receiver locations as a time series. The aim of seismic imaging is to reconstruct the subsurface elastic properties from the immense dataset that is collected in the field. This challenge was initially taken up by global seismologists who estimated a large scale and smooth distribution of the elastic earth properties using seismic traveltimes (Gutenberg, 1951; Kronig et al., 1946). Amplitude data was not reliable, as the receivers available were not calibrated properly to record true amplitudes. This method worked well when offsets were large and desired resolution was low. The sensitivity of traveltimes to changes in velocities could be calculated by brute force finite difference methods or simple analytic expressions as the number of model parameters was small.

Exploration Seismology works on a completely opposite paradigm where the onus is on high-resolution imaging. Traveltime based tomographic models do not have the resolution needed for hydrocarbon exploration and drilling decisions. Source and receiver grids used in exploration experiments are denser

compared to those used in global seismology making gradient computation using the brute force approach impractical. Estimating high-resolution interface properties became a priority and therefore amplitude summation from the exploding reflector model was the main principle behind migration which provided detailed images of the subsurface (Claerbout, 1971, 1985).

Because exploration seismology acquires limited offsets, seismic imaging is insensitive to the intermediate wavelengths (Jannane et al., 1989). This led to the development of a two-step workflow where the macro-model is first constructed using traveltime information and the amplitudes are focused using different types of migration techniques. (Claerbout and Doherty, 1972; Gazdag, 1978; Stolt, 1978). The procedure proved to be very effective in simple geological models but achieved limited success in complex models like salt domes, sub-salt targets, foothills and thrust belts. A few approaches involving updating the macro-model have been proposed by Snieder et al. (1989) and Docherty et al. (1997), but they are plagued by frequency and offset limits.

Lailly (1983) and Tarantola (1984) re-formulated the migration imaging theorem (Claerbout, 1985) as a local optimization problem which aimed to reduce data misfit between observed and modeled data. They showed that the gradient of the misfit function can be estimated by cross-correlating the incident wavefield from the source with the backward propagating residual wavefield. This operator was identified to be the equivalent of migration, and when this gradient is added to the initial velocity model, the perturbations lead to an updated velocity model which forms the starting model for the

next iteration. This approach is expected to work better than conventional migration because the gradient is calculated using all available data in the seismogram summed over all shots and receivers.

Several authors including Crase et al. (1990),Djikpéssé and Tarantola (1999) and Pica et al. (1990), used waveform fitting migration on reflectivity and ray tracing data. However, the applications of these methods were limited because they were computationally intensive at that time. As computation power increased it became feasible to use this waveform fitting technique on a regular basis, the accuracy of which depended on the accuracy of the modeling kernel. When the full wave equation is solved for generating synthetic seismograms during the least square optimization, the process is known as Full Waveform Inversion (FWI). The benefit of FWI is that it includes all types of waves in the optimization including reflected waves, refracted waves, turning rays, diverging rays and super critical rays (e.g Sen and Roy (2003)). The techniques for modeling may be finite difference (Virieux, 1986) finite element (Marfurt, 1984) or pseudo-spectral (Danecek and Seriani, 2008) but the basic inversion framework remains the same as described in Lailly (1983) and Tarantola (1984).

1.1 Background Theory

This thesis deals primarily with the acoustic wave equation but some of the methods can be extended to anisotropic and elastic formulations. Without loss of generality, we discuss the FWI formulation in its constant density wave

acoustic formulation in both time and frequency domain.

1.2 Full Waveform Inversion

Full Waveform Inversion (Pratt and Worthington, 1990; Virieux and Operto, 2009) is an iterative local optimization algorithm that updates a starting model using the gradient of the misfit function. The initial model is normally a model which explains the kinematic characteristics of observed data which FWI modifies incrementally over several iterations to match amplitude and phase characteristics. A simple flowchart of the algorithm is shown in Figure 2.4. Essential elements of FWI are discussed below.

1.2.1 Forward Modeling

I use the acoustic approximation and solve the constant density wave equation given by

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 P(\mathbf{x})}{\partial t^2} = \nabla^2 P(\mathbf{x}) + s\left(\mathbf{x}, t\right), \tag{1.1}$$

where P is the pressure wavefield, ∇^2 is the Laplacian given by $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}$, $v^2(\mathbf{x})$ is the velocity field and $s(x,t)$ is the source term.

In the frequency domain, it is given by

$$\nabla^2 P(\mathbf{x}) + \frac{\omega^2}{v^2(\mathbf{x})} P(\mathbf{x}) = s(\mathbf{x}, \omega), \tag{1.2}$$

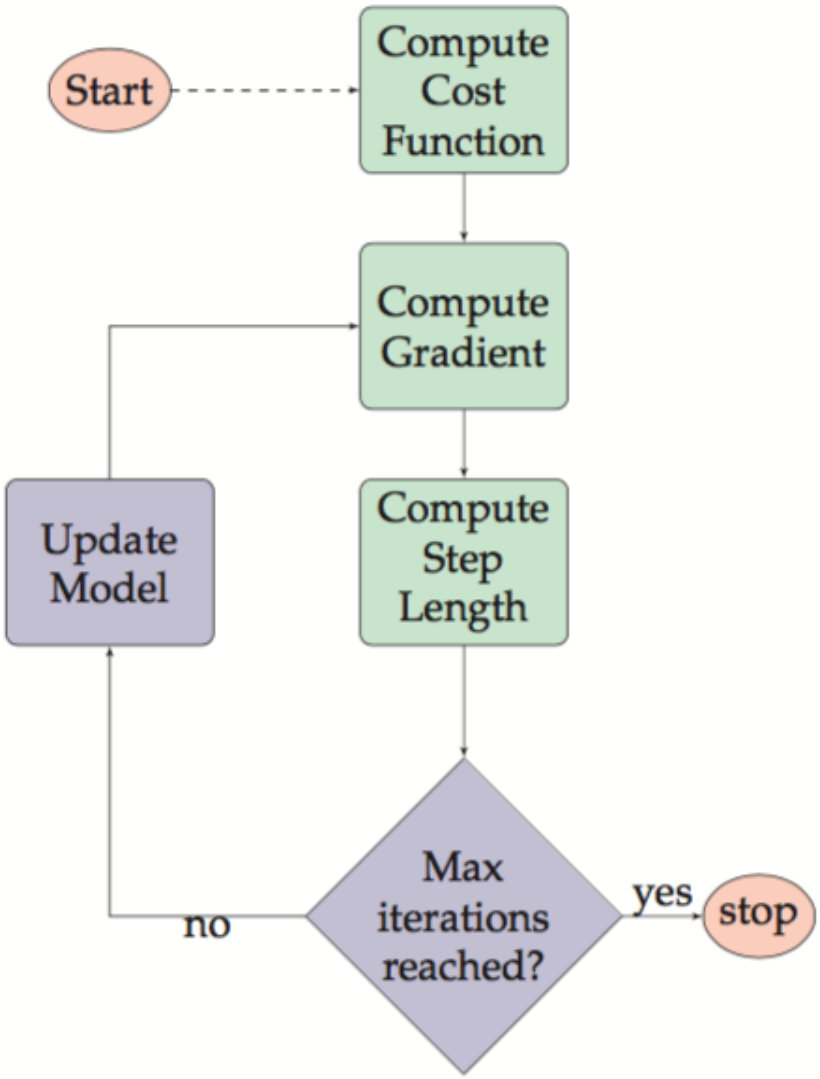


Figure 1.1: Flowchart showing the Full Waveform Inversion algorithm

where, $P(\mathbf{x})$ is the pressure wavefeld, $\mathbf{x}$ is the co-ordinate vector, $v(\mathbf{x})$ is the P wave velocity $s(\mathbf{x}, \omega)$ is the source, and ω is the angular frequency.

The wavefield is sampled at the receiver locations over all time steps to obtain the seismogram.

1.2.2 Cost Function

The cost function is the L2 norm of the difference between observed and synthetic data which is given in time domain as

$$E = ||\mathbf{d_{obs}} - \mathbf{d_{syn}}||_2, \tag{1.3}$$

where $\mathbf{d_{obs}}$ is the observed data and $\mathbf{d_{syn}}$ is the synthetic data.

1.2.3 Gradient Computation

I use the adjoint state method after Plessix (2006) to compute the gradient, which is given by the zero-lag cross-correlation of the forward propagating source and the backward propagating residual. The adjoint wavefields are computed by back propagating the data residuals in time using

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 R}{\partial t^2} = \nabla^2 R + \Delta d\left(\mathbf{x}, t\right), \tag{1.4}$$

where R is the adjoint wavefield , ∇^2 is the second order laplacian, $\Delta d\left(x, t\right)$ is the data misfit.

The frequency domain equivalent of the adjoint wavefield propagation is given by

$$\nabla^2 R(\mathbf{x}) + \frac{\omega^2}{v^2(\mathbf{x})} R(\mathbf{x}) = \Delta d(\mathbf{x}, \omega), \tag{1.5}$$

where, R is the adjoint wavefeld.

The gradient is computed by a zero-lag crosscorrelation of the forward and adjoint wavefields. In the time domain, it is given by

$$\frac{\partial E}{\partial v(\mathbf{x})} = \frac{-2}{v\left(\mathbf{x}\right)^3} \sum_{shots} \ddot{P}\left(\mathbf{x}\right) R\left(\mathbf{x}\right), \tag{1.6}$$

where $\mathbf{P}$ and $\mathbf{R}$ are the source and adjoint wavefields in space and time. The $\ddot{}$ operator denotes second derivative in time.

In the frequency domain, it is given by

$$\frac{\partial E}{\partial v(\mathbf{x})} = Re \sum_{\mathbf{x}_{shots}} \frac{-2\omega^2}{v(\mathbf{x})^3} R(\mathbf{x}) P(\mathbf{x}) \tag{1.7}$$

where, Re denotes the real part.

1.2.4 Optimization

Once the gradient has been computed, one can use it in a minimization loop using Steepest Descent, Conjugate gradient Vigh and Starr (2008), L-BFGS (Asnaashari et al., 2013a) or Newton methods (Métivier et al., 2013).

The step length is computed using cubic or parabolic methods which require 2 more forward evaluations. As a result, one iteration of gradient-based

FWI requires one forward and one backward evaluation of the wave equation to compute the gradient. Additionally, it requires 2 or more forward model evaluations to estimate a step length. Therefore, there are at least 4 modeling operations required for one iteration of FWI. The reader is referred to Virieux and Operto (2009) for an excellent overview of FWI.

1.3 Limitations

1.3.1 Cycle Skipping

Although FWI is well established in theory, it is still a local optimization algorithm which means the starting model must to be very close to the optimal model to obtain convergence. Otherwise, the inversion converges to a local minimum, giving an inaccurate representation of the subsurface. This is because the misfit function has multiple local minima and a local optimization method drives the solution to the closest local minimum. The global minima trough spans half wavelength across each event peaks or toughs. This phenomenon is called cycle skipping; Virieux and Operto (2009) explain how this proximity translates from model domain to data domain as shown in Figure 1.2. Let us consider the solid trace in the middle as the observed trace. There are two trial traces on top and bottom of the observed trace, which I need to optimize to match to the observed trace. To match my trial traces to the observed traces in the data domain, the corresponding peak of the trial trace has to be within half wavelength of the observed trace. The zone of half wavelength is shown in grey where the peak of the trial traces must reside for the

inversion to converge. If it is outside the zone, the algorithm will converge to a model which is not the optimal solution. it will converge to the nearest local minima giving an incorrect estimation of the subsurface properties.

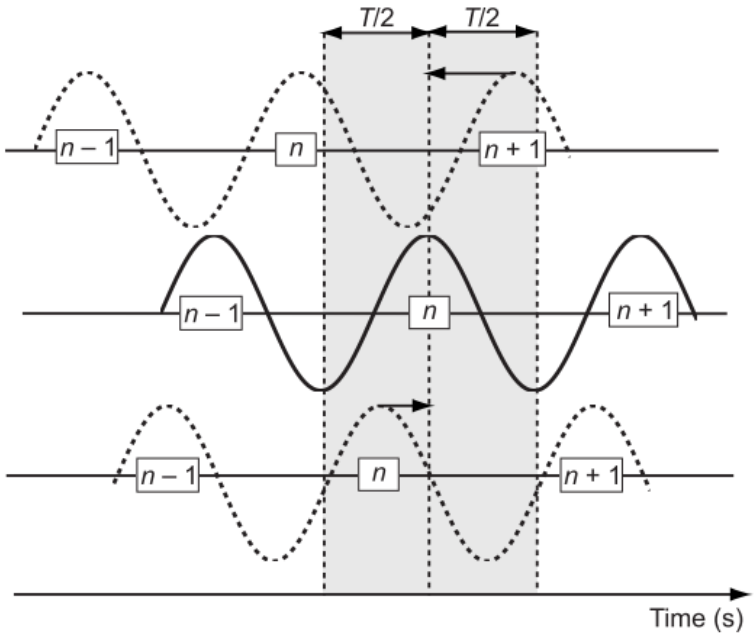


Figure 1.2: Schematic of cycle-skipping artifacts in FWI after Virieux and Operto (2009). The solid black line represents a monochromatic seismogram of period T as a function of time. The upper dashed line represents the modeled monochromatic seismograms with a time delay greater than T/2. In this case, FWI will update the model such that the $n+1th$ cycle of the modeled seismograms will match the nth cycle of the observed seismogram, leading to an erroneous model. In the bottom example, FWI will update the model such that the modeled and recorded nth cycle are in-phase because the time delay is less than T/2.

1.3.2 Computational issues

Several numerical methods are available for solving the wave equation for use in FWI. These include Finite Difference (Crase et al., 1990; Levander, 1988; Virieux, 1986) , Finite Element (Choi et al., 2008), psuedo-spectral (Wu and Alkhalifah, 2014) or spectral-element methods (Capdeville et al., 2005; De Basabe and Sen, 2007; Magnoni et al., 2014) both in time domain (Crase et al., 1990; Tarantola, 1984) or in the frequency domain (Pratt and Worthington, 1990). In all approaches the common feature is an expensive modeling/adjoint operation, which needs to be solved multiple times to update the starting model. This makes FWI prohibitively expensive, especially in 3D.

Additionally, FWI suffers from the data reversal problem in the time domain. The order of generation of the forward wavefields is opposite to its access when computing the gradient. In the presence of limited memory, this necessitates checkpointing (Griewank and Walther, 2000) which further increases the computational cost in FWI. This issue is especially critical in 3D where the cost of wavefield reconstruction can be several times more than generating them.

1.4 My Contributions

The main contributions of this thesis are as follows:

- Development of a gradient-free optimization method to obtain a good starting model for gradient-based FWI and its applicability in 2D and

3D.

- Development of a fast forward modeling method in the frequency domain.
- Development of a new checkpointing algorithm for adjoint problems in dedicated processing units like GPUs.

1.5 Previous Work

Several attempts have been reported in the literature to address the issues of computational speed and cycle skipping problem in FWI. I discuss those in individual chapters so that they are self-contained.

1.6 Outline

This thesis can be broadly classified in two parts. Part 1 contains 4 chapters and deals with the issue of starting model in FWI. They are described as follows

- Chapter 2 proposes a global optimization method for 2D FWI using sparse parameterization. Sparse parameterization is done using velocities and interfaces and minimized using a Very Fast Simulated Annealing approach.
- Chapter 3 extends the 2D approach in Chapter 1 to include estimation of complex salt bodies using a combination of global optimization, pattern recognition & level set methods.

- Chapter 4 extends the 2D approach described in chapter 2 to 3D FWI using the same overall approach but with a different sparse parameterization.

- Appendix A introduces an alternate swarm based optimization technique called Particle Swarm Optimization to estimate starting models without a good starting guess.

In part 2, I describe my development of methods to reduce the computational cost of modeling and gradient in FWI. The chapters in part 2 are as follows

- Chapter 5 describes how to exploit unchanging zones in a velocity model and pre-computed response in subsequent modeling evaluations in FWI. The method is based on the Schur Complement approach and applies only to Frequency Domain methods.

- Chapter 6 describes a data-centric approach to gradient computation in GPUs using a novel two-level checkpointing method. The method is applicable to any adjoint problem and is parameterizable to compute performance and bandwidths.

Finally, Chapter 7 summarizes the contribution of this thesis and possible avenues for future work.

Part I

Global Optimization in FWI

Chapter 2

Estimating a starting model for full waveform inversion using a global optimization method

2.1 Summary

Full waveform inversion (FWI) has become a popular method to estimate elastic earth properties from seismograms. It is formulated as a data fitting least-squares minimization problem, which iteratively updates an initial velocity model with the scaled gradient of the misfit until a satisfactory match between real and synthetic data is obtained. However, such a local optimization approach can converge to a local minimum if the starting model used is not close enough to an optimal model. I propose a two-step process in which I first estimate a starting model using a global optimization method. Unlike local optimization methods, a global optimization method starts with a random starting model and is not generally susceptible to be trapped in a local minimum. The starting model for FWI that I aim to estimate is sparsely parameterized and contains a set of interfaces and velocities, which are used

Parts of the chapter were first published in *Datta, D., & M. K. Sen. Hybrid optimization methods for full waveform inversion, AGU Fall Meeting Abstracts, 2014.* The peer-reviewed version of this paper can be found in *Datta, D., & Sen, M. K. (2016). Estimating a starting model for full-waveform inversion using a global optimization method. Geophysics.*. This work was done by Debanjan Datta under his advisor's guidance.

to represent the entire velocity model. I obtain the depth of the interfaces and the velocities by minimizing the data misfit in the least-square sense using a global optimization method called Very Fast Simulated Annealing (VFSA). Once the sparse velocity model is obtained from VFSA, I use that as a starting model in a conventional gradient-based FWI to obtain the final model. I apply the proposed method to one synthetic dataset and two field datasets from offshore India. The proposed method is able to extract a starting model in each case, which satisfactorily converges to an optimal solution that is in good agreement with the available well log and predicts the observed data very well. I demonstrate that with proper choice of model parameterization and optimization parameters, one is able to take advantage of VFSA and gradient descent methods. The resulting algorithm is found to be fairly fast and accurate for practical implementation of FWI.

2.2 Introduction

Determining subsurface structure and properties is the ultimate aim of seismic exploration. Out of a plethora of algorithms, full waveform inversion (FWI) (Tarantola, 1984; Virieux and Operto, 2009) is considered the most accurate velocity analysis technique with a theoretical resolution of half the propagating wavelength. It is set up as a local optimization problem where the initial model is iteratively updated with a scaled gradient of the objective function until there is a satisfactory match between the observed and synthetic data. FWI can be divided into three main stages of individual operations. The

first step is to simulate the seismic response of an earth model (forward modeling) and compute the residuals between the synthetic and observed data. In most cases, the finite-difference method is used (Crase et al., 1990), but finite-element (Choi et al., 2008), spectral wavefield extrapolation (Wu and Alkhalifah, 2014), and spectral-element methods (Capdeville et al., 2005; De Basabe and Sen, 2007; Magnoni et al., 2014) are also common. In the second step, the gradient of the misfit function is computed by summing the zero-lag correlation of the forward-source propagated wavefield and back-propagated residual wavefield (Chavent and Jacewitz, 1995) over shots and time steps. The adjoint state method estimates the gradient of the misfit function using only two forward calculations making it computationally feasible. The final step consists of updating the current model with a scaled and conditioned gradient using a local optimization method like steepest descent, conjugate-gradient or a pseudo-Newton method. Forward modeling and gradient calculations can be performed in either the time (Crase et al., 1990; Tarantola, 1984) or frequency domain (Pratt and Worthington, 1990). Virieux and Operto (2009) provide a summary of these methods.

Because FWI is a local optimization algorithm, it is sensitive to the choice of the starting model. The starting model needs to be close to the true model for optimal convergence. If the starting model is not close to the true model, it converges to a local minimum closest to the starting model giving an incorrect estimation of the subsurface properties as well as inadequate data fit. The velocity models used in FWI contain two kinds of information 1)

tomographic component or low wavenumber content that affects the traveltimes and 2) migration component or high wavenumber content that affects amplitudes rather than traveltimes.

To obtain a high-resolution and accurate velocity model characteristic of FWI, both the tomographic and the migration components need to converge (Mora, 1989). However, because of the greedy (moving in the direction of decreasing error) nature of the algorithm, the migration component can only converge if the tomographic component is in agreement with the observed data within half the dominant period of the propagating wave. This is because even small changes in the low-frequency content in the model affect the traveltimes by a significant amount, leading to a phenomenon called cycle skipping explained in Figure 2.1. In practice, the tomographic component of the velocity model is not estimated using FWI but is provided as a starting guess. This starting guess is often derived from conventional velocity analysis techniques like traveltime tomography or interval velocities. Fixing the low-frequency starting model avoids the task of estimating the non-linear low-frequency component, but the accuracy of the starting model still depends on conventional velocity analysis techniques.

Several authors have proposed different methods to address the issue of local minima as well as stability to noise in FWI. These methods can be broadly divided into three classes. The first class comprises the multi-scale technique (Bunks et al., 1995; Sirgue and Pratt, 2004) where the velocity model is built over multiple inversion runs by staging over frequencies. The

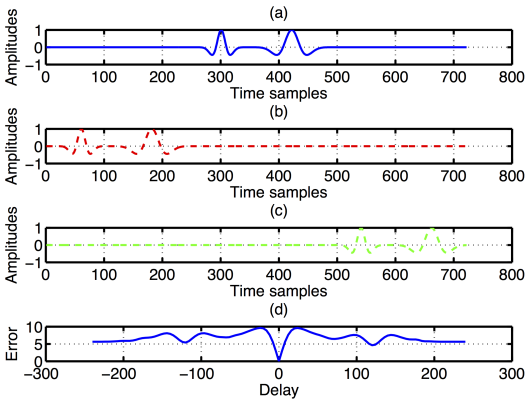


Figure 2.1: A schematic showing the phenomenon of cycle skipping in FWI. (a) shows an observed trace one wishes to match. (b) and (c) show two trial traces shifted by different time samples compared to (a). Using (b) and (c) as limits on either side, a misfit vs shift curve is shown in (d) where the presence of several local minima troughs can be seen. These local minima troughs can be avoided if the observed and trial traces are in agreement within half the propagating wavelength of the wave. If not, the convergence happens in a local minimum.

final model from a limited frequency band inversion is used as the starting model for the next frequency band. Thus, higher frequencies are added after each frequency band inversion mitigating the cycle skipping problem to a large extent. Dessa et al. (2004) and Fichtner et al. (2013) applied the multi-scale method to continental and regional seismic data, respectively. Brossier et al. (2009) applied it to exploration data consisting of both synthetic and real data sets.

The second class of algorithm consists of modifying the objective function in FWI to reduce the multi-modality of the objective function so that the inversion is not plagued by the presence of multiple local minima. The Laplace

domain FWI formulation was proposed by Shin and Min (2006) where damped wavefields are used to constrain the objective function to a very wide global minima well. The damping allows estimating low wavenumber models with more robustness that can be used as starting models for FWI. This work was extended by Kim et al. (2013) to the Laplace Fourier domain FWI where both high and low wavenumbers could be estimated. Very recently Almomin et al. (2012) proposed Tomographic FWI where they proposed a composite misfit function with a term focused on long wavelengths. This long wavelength term introduces the tomographic component avoiding cycle skipping. Deconvolution based approaches also fall in this category in which large time shifts are imparted into the modeled traces to avoid cycle skipping. A recent example of deconvolution based method is Warner and Guasch (2014), that uses Wiener filters to shift seismic events to avoid cycle skipping. Ma and Hale (2013) use Dynamic Image Warping (DIW) to impart time shifts to match seismic events between an observed and modeled trace providing a robust method of estimating the low wavenumber components of the model.

The third category of methods are the image domain approaches. These approaches extend the model to improve model-data correlation, and a measure of semblance across different gathers is implemented in the objective function. These have been demonstrated using Differential Semblance Optimization (DSO) (Shen and Symes, 2008; Symes and Carazzone, 1991) and wave-equation migration velocity analysis (Sava and Biondi, 2004a,b). However, image domain methods are primarily focused on reflection arrivals and

are not very robust when dealing with turning rays.

Here, I propose a two-step method where the starting model for FWI is first estimated using a global optimization method called Very Fast Simulated Annealing (Afanasiev et al., 2014; Varela et al., 1998; Xia et al., 1998a) coupled with a sparse parameterization method, then proceeding with a conventional gradient-based FWI using the inverted model obtained after VFSA. Sparsity in the global optimization method is enforced by inverting for a set of interfaces and velocities rather than the complete velocity model. Thus, the objective here is not to match an entire trace but just the major events. The resulting model is expected to contain all major interfaces with sharp velocity jumps. This model is then smoothed slightly to remove the sharp velocity jumps and used as the starting model for the standard gradient-based FWI, which converges to an accurate velocity model that agrees very well with the observed seismic data. In the following sections, I provide a theoretical overview of FWI and explain the theory behind my proposed method. I demonstrate this method using one synthetic example and two real data sets from offshore India. The proposed method is able to estimate accurate velocity models, which are very close to the true velocity model. The models for real data show good agreement with seismic data and available well logs.

2.3 Theory

My proposed approach has two distinct parts:

1. Estimation of a starting model for FWI using VFSA, and

2. Use of smoothed final model from VFSA as the starting model in a gradient-based FWI.

These steps are discussed below.

2.3.1 Estimation of a starting model using VFSA

I aim to estimate a starting model by minimizing the data misfit between observed and synthetic data. However, it is important to note that inverting for a velocity model in a finite difference computational grid using a global method is generally intractable because of a large number of model parameters. Inverting for such a dense set of parameters would be computationally prohibitive due to slow convergence and non-uniqueness. Therefore, to stabilize the inversion and obtain fast convergence, I propose a sparse parameterization technique to reduce the number of model parameters. The sparse parameterization technique represents a gridded velocity model using a set of interfaces and two velocities associated with each interface (Sen and Frazer, 1985; Zelt and Smith, 1992). Every interface is represented by a set of nodes represented by (x_k^i, z_k^i) where x_k^i and z_k^i are the x (lateral position) and z (depth) coordinates of the ith point in the kth interface, respectively. Unlike Sen and Frazer (1985), who used arc-tangent functions for interpolating interfaces across nodes, I use constrained non-oscillatory splines (Kruger, 2003) for this purpose. Non-oscillatory splines are preferred over normal cubic splines because non-oscillatory splines do not show unwanted oscillations compared to normal cubic splines. This effect is shown in Figure 2.2 where I compare

the interpolation performance of both oscillatory and non-oscillatory splines using the same set of nodes. The interpolated results show unwanted oscillations with cubic splines that are not observed when non-oscillatory splines are used. Therefore using non-oscillatory splines not only prevents unwanted oscillations but also allows representation of the same interface using fewer points than normal cubic splines. Each interface has two velocities associated with it, namely, v_{up}^k and v_{dn}^k, the velocities just above and below the kth interface, respectively. After the interfaces have been interpolated using non-oscillatory splines and the velocities are fixed in the immediate vicinity of the interfaces, the velocities in between interfaces are linearly interpolated in the vertical direction. I employ two constraints when inverting for interfaces and velocities using VFSA. Firstly, two interfaces are not allowed to cross each other, and secondly, if v_{up} and v_{dn} for an interface come in within 5 percent of each other, the specific interface is eliminated. Using these constraints allow the algorithm to predict the correct velocity if the apriori interface information does not have an interface in the real world. An example velocity model with my proposed parameterization is shown in Figure 2.3, and the flowchart of my sparse parameterization can be seen in Figure 2.4.

My global optimization problem solves for the co-ordinates of the nodes (x_k^i, z_k^i) and (v_{up}^k, v_{dn}^k) to minimize the misfit between the observed data and synthetic seismograms from the velocity model generated by my proposed sparse parameterization. The misfit function used in VFSA is the cross correlation objective function (Sen and Stoffa, 1991) given by

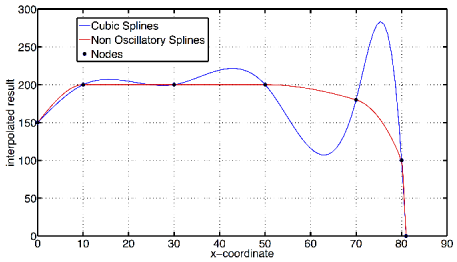


Figure 2.2: Figure showing the performance of cubic (Blue) and non oscillatory splines (Red): The black dots are the nodes used for interpolation and the blue and red curves show the result of spline interpolation and non oscillatory interpolation respectively. Cubic splines show considerable oscillations which are not present in non-oscillatory splines. This allows representation of an interface more accurately using fewer points.

$$E_{VFSA} = -\frac{\mathbf{d}_{obs} \otimes \mathbf{d}_{syn}}{(\mathbf{d_{obs}} \otimes \mathbf{d}_{obs})^{\frac{1}{2}} (\mathbf{d}_{syn} \otimes \mathbf{d}_{syn})^{\frac{1}{2}}}, \tag{2.1}$$

where, $\mathbf{d}_{obs}$ is the observed data and $\mathbf{d}_{syn}$ is synthetic data and $\otimes$ represents correlation. Note that this function is not sensitive to absolute amplitudes in the observed and synthetic seismograms.

In this paper, I use the acoustic approximation and solve the constant density wave equation given by

$$\frac{1}{\mathbf{c}(x,z)^2} \frac{\partial^2 \mathbf{P}}{\partial t^2} = \nabla^2 \mathbf{P} + s\,(x,t)\,, \tag{2.2}$$

where $\mathbf{P}$ is the pressure wavefield, ∇^2 is the Laplacian given by $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2}$,

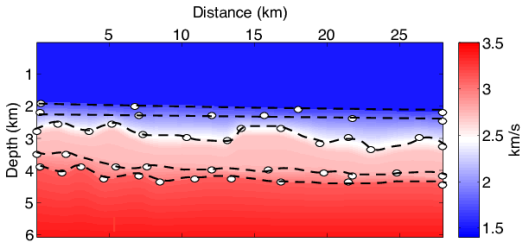


Figure 2.3: An example velocity model derived from the proposed parameterization. I build the velocity model using five interfaces. The nodes in each interface are shown with white circles while the dotted black line show the interface after interpolation. The velocities v_{up}^k and v_{dn}^k are fixed across the interfaces and the velocities between interfaces are estimated using linear interpolation.

$\mathbf{c}(x, z)$ is the velocity field and $s(x, t)$ is the source term. I solve equation 2.2 using an 8th order in space and 2nd order in time finite difference method to compute the wavefields at each time step. The wavefield is sampled at the receiver locations over all time steps to obtain the seismograms. Before I explain the inversion process, a brief theoretical overview of VFSA is presented below.

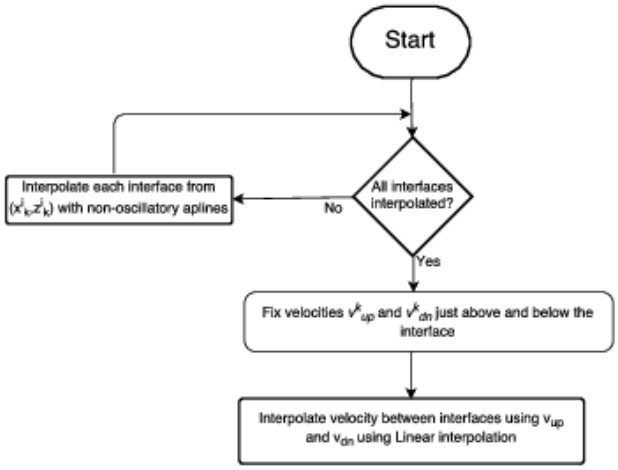


Figure 2.4: A flowchart of the proposed sparse parametrization method using non oscillatory splines for VFSA inversion to obtain a good starting model.

2.3.2 Very fast simulated annealing

VFSA (Ingber and Rosen, 1992; Sen and Stoffa, 2013) is an improved version of simulated annealing used for minimizing an objective function starting from a random guess. It converges faster than conventional simulated annealing, which is inspired by statistical physics and mimics the process of slow cooling of a heated material. As a heated material cools down, the energy of the molecules decreases. However, the final configuration of the atoms depends on the rate of cooling. Rapid cooling or quenching does not allow atoms many degrees of freedom to orient themselves resulting in an amorphous condensate. Slow cooling or annealing allows atoms sufficient degrees of freedom to orient themselves in a structure resulting in a crystalline material having the lowest

possible energy state.

This property is exploited in SA where one draws an analogy of the energy level with the objective function. The cooling is simulated using a dimensionless parameter called temperature, and the model evolution is controlled by the Metropolis criterion (Metropolis et al., 1953). The starting model $\mathbf{m}$ is perturbed by a random step $\delta\mathbf{m}$ to derive a new trial model $\mathbf{m_{new}}$, and their respective objective functions are calculated. The new model is selected from a gaussian proposal distribution in the conventional simulated annealing and from a Cauchy distribution in VFSA. The Cauchy distribution allows for different distributions depending on the temperature that is lowered with each iteration. Therefore, in early iterations, VFSA is likely to choose a trial model far from the current model while at later iterations, it will likely search in the close neighborhood of the current model. Use of this temperature-dependent proposal distribution speeds up convergence of simulated annealing. If $E(\mathbf{m}_{old}) \geq E(\mathbf{m}_{new})$, then new model is accepted. If $E(\mathbf{m}_{old}) < E(\mathbf{m}_{new})$, the uphill model is accepted with a probability Pr given by

$$Pr = e^{(-\Delta E/T)}, \tag{2.3}$$

where T is temperature, and $\Delta E = E(\mathbf{m}_{new}) - E(\mathbf{m}_{old})$. From equation 2.3, one can see that at higher temperatures, worse solutions are more likely to be accepted than at lower temperatures; This allows the algorithm to accept worse solutions at high temperatures and to reject them at low temperatures.

Typically, VFSA needs anywhere between 100 to 1000 iterations to converge for a well conditioned optimization problem (See Sen and Stoffa (2013) for examples). I started with a temperature of $T_0 = 500$ and used a decay function of $T_n = T_0 0.98^n$ with n iterations. For the problems shown in this paper, I obtained convergence in about 200 iterations. A more detailed explanation for the choice of number of iterations is provided in the discussions section.

2.3.3 Gradient FWI from VFSA model

The final model from VFSA contains sharp velocity changes across interfaces that are not desirable in a starting model for FWI. Therefore, to let the gradient FWI adjust interfaces in the final model, I smooth the starting model obtained from VFSA before using it as the starting model for FWI. The objective function for this stage, unlike in VFSA, is a L2 norm based misfit (observed-synthetic data) given by

$$E_{grad} = ||\mathbf{d_{obs}} - \mathbf{d_{syn}}||_2. \tag{2.4}$$

The reader is referred to section 1.2 for a theoretical overview of FWI. I use the time domain implementation in this chapter.

Minimizing equation 2.4 in the least squares iterative sense might lead to oscillatory and non-feasable solutions because of the non-linearity of the misfit function. To mitigate this, I stabilize the inversion by using Tikhonov

regularization, which adds an additive term of the second derivative of the model. The modified objective function (Asnaashari et al., 2013a) is given by

$$E = \|\mathbf{d}_{obs} - g(\mathbf{m})\|^2 + \lambda \frac{1}{2} \left\{\mathbf{m}^T \mathbf{D} \mathbf{m}\right\}^{1/2}, \tag{2.5}$$

where λ is a weight for the regularization term, and $\mathbf{D}$ is the second order Laplacian operator. I follow similar techniques to find λ as mentioned in Sen and Biswas (2015).

I use a conjugate gradient method (Polak and Ribiere, 1969) in my FWI formulation where the model is iteratively updated as

$$\mathbf{m}_{i+1} = \mathbf{m}_i - \alpha_i \mathbf{h}_i, \tag{2.6}$$

where $\mathbf{m}_{i+1}$ is the model in $(i+1)th$ iteration and α is the step length, and $\mathbf{h}_i$ is the step direction given by Vigh and Starr (2008) as.

$$\mathbf{h}_0 = \mathbf{g}_0, \tag{2.7}$$

$$\beta_i = \frac{\mathbf{g}_i^T (\mathbf{g}_i - \mathbf{g}_{i-1})}{\mathbf{g}_{i-1}^T \mathbf{g_{i-1}}}, \tag{2.8}$$

$$\mathbf{h}_i = \mathbf{g}_i + \beta_i \mathbf{h}_{i-1}, \tag{2.9}$$

where $\mathbf{g}_0$ is the initial gradient given by equation 1.4 . The step length α_i is computed in every iteration using a quadratic line search algorithm (Wright and Nocedal, 1999).

2.4 Results

I apply my proposed method to one synthetic example and two field datasets from offshore India. In all my examples I perform VFSA and inversion in the time domain and the full data bandwidth was inverted in one pass. I use absorbing boundary conditions on all four boundaries.

Synthetic Data

For the synthetic test, I use the SEG-EAGE Overthrust model shown in Figure 2.5a. The model contained 187 and 801 grid points in z and x directions, respectively, with a grid spacing of 25m. Two hundred sources were placed every 100m over the length of the model, and receivers were laid every 50m over the length of the model. I generate synthetic data using a 15Hz Ricker wavelet at 200 shot locations that are placed 100 meters apart. From the generated shot gathers, I display a zero-offset section where I pick 4 interfaces as shown in Figure 2.5b. The interfaces are picked by visual inspection and as I will demonstrate in the discussions section of the text, this approach works well enough and the final model is fairly consistent. The interfaces are picked in time, which are then converted to depth using the mean velocity of the search space given by $(v^k_{max}+v^k_{min})/2$ where v^k_{min} and v^k_{max} are the min and max of the search space, respectively. It is important to note that the initial guess need not be accurate to begin with because the solution would be updated by VFSA avoiding local minima. However, the mean approach offers a good balance of model exploration and exploitation while facilitating fast convergence. This

is shown in Figure 2.6a where I plot a random starting model for VFSA and observe that it looks very different from the final result shown in Figure 2.6c. The total number of nodes required for the parameterization was 28 for 4 interfaces and 8 velocities. I filtered the data using a 10Hz lowpass filter and ran my VFSA algorithm to obtain optimal positions of the interfaces. The final model derived after VFSA inversion is shown in Figure 2.6b. The velocity model agrees with all major interfaces and provides a good starting model for gradient FWI. I use the smoothed model from Figure 2.6b as the starting model for a conventional gradient-based FWI. The full data bandwidth was inverted in one pass, and the final model obtained is shown in Figure 2.6c after 80 iterations. The final velocity model is in good agreement with the true model (Figure 2.5a). I compare the seismograms (10Hz) from the true model, the VFSA model and FWI model for a shot located at $x = 100m$ in Figure 2.7. For further clarity, I plot the traces from the true (red), VFSA (blue) and FWI (black) models at offsets of 2.5km, 7.5km and 10km in Figure 2.8. It can be seen that the events from the VFSA model are within half the dominant period of the seismogram obtained from the true model. Thus this method is able to ensure the kinematic match between the true and starting models for FWI, which allows for optimal convergence. There is some mismatch that is compensated by performing gradient FWI.

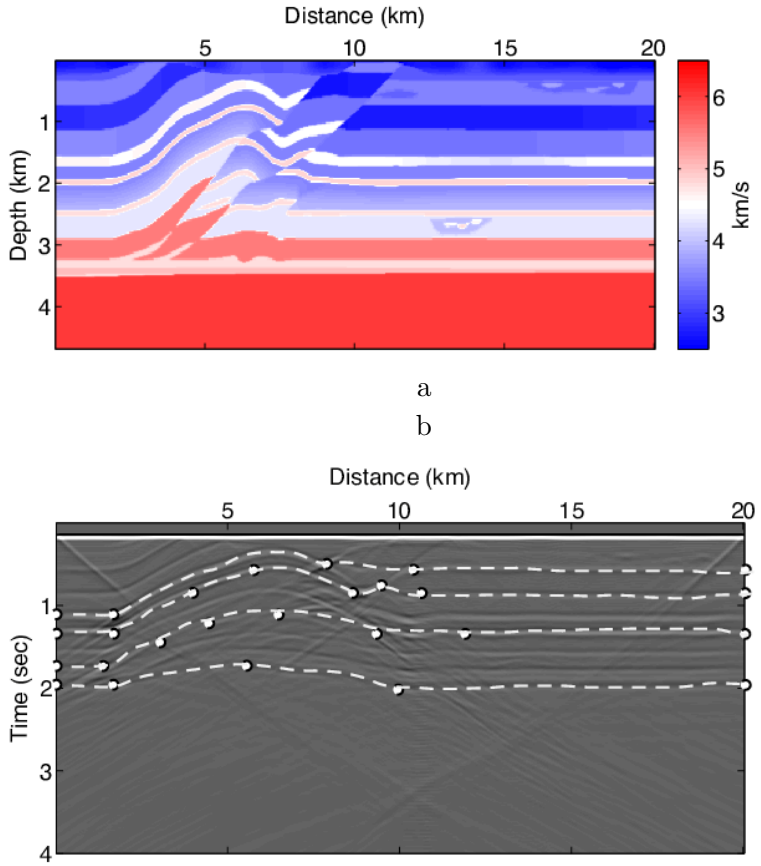
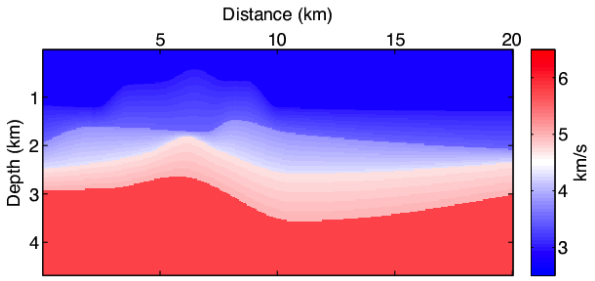
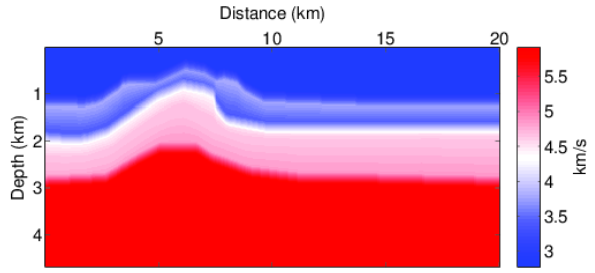


Figure 2.5: (a) The true SEG-EAGE Overthrust model. (b) A strategy for fixing the search space of VFSA parameters. I generate a zero offset section of the SEG-EAGE Overthrust model and pick 4 interfaces. I pick nodes for interfaces in time following seismic events. The picked nodes are shown by white circles and the events are shown by dotted white lines.

a



b



c

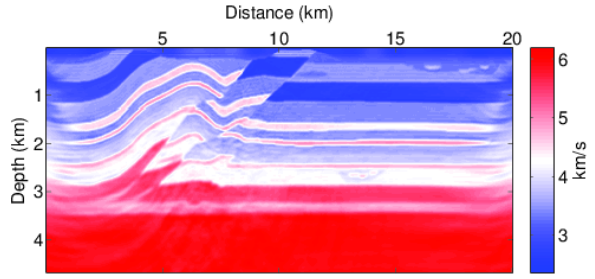


Figure 2.6: (a) Figure showing a random starting model for VFSA. The interfaces in the starting model are completely random and do not show any resemblance to the final model after VFSA shown in (b). (b)The velocity model obtained after 200 iterations of VFSA using 4 picked interfaces from Figure 2.5b. The estimated velocity model is fairly smooth with a few marked interfaces. (c) The velocity model obtained after 80 iterations of gradient-based FWI using the model from a smoothed version of (b) as the starting model. This velocity model is able to retrieve all high wavenumber features of the velocity model and is very similar to the true model shown in Figure 2.5a.

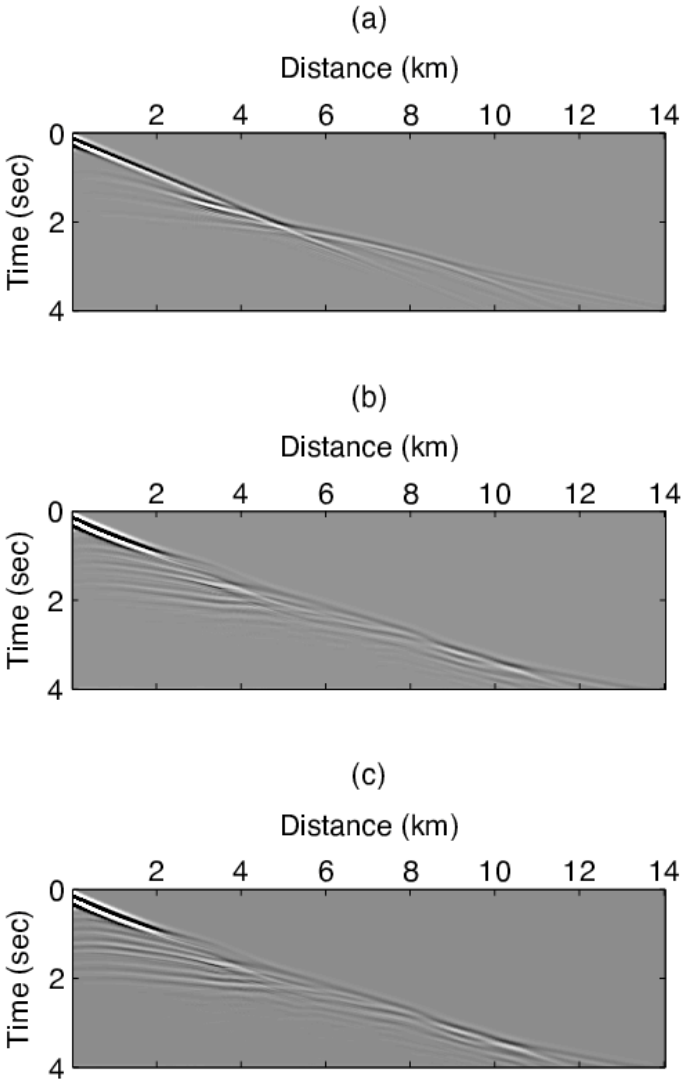


Figure 2.7: Comparison of synthetic seismograms for a single shot at x=100m for the SEG-EAGE Overthrust model. (a) shows the seismogram generated from the model generated by VFSA. (b) shows the seismogram obtained from the final model after FWI and (c) shows the seismogram from the true model. Trace comparison of a few offsets are shown in Figure 2.8

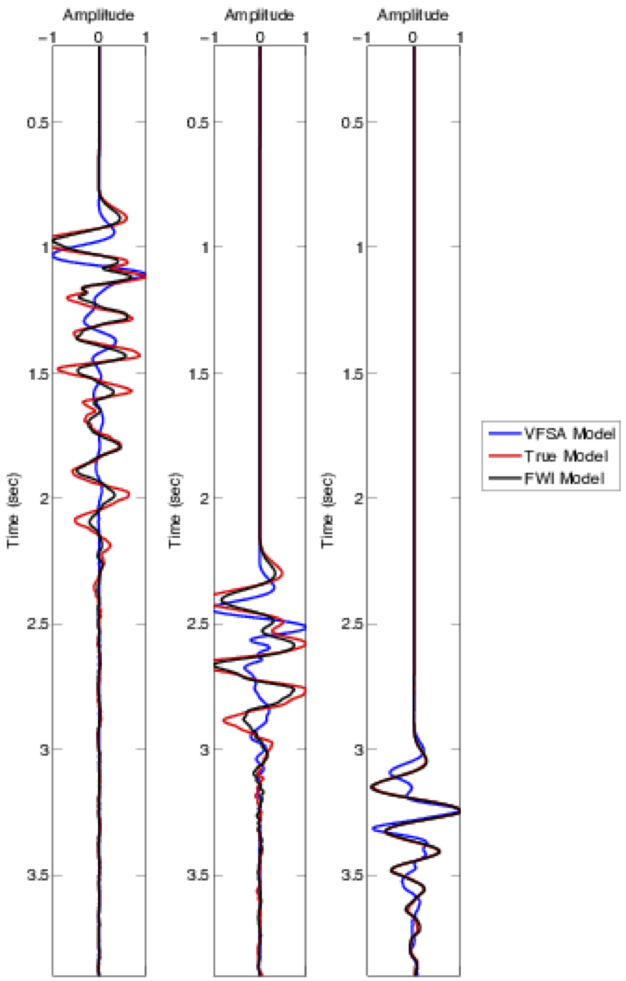


Figure 2.8: Comparison of traces between the seismograms (at 10Hz) computed from the true model(red), VFSA model (blue) and FWI model (black) at different offsets in Figure 2.7. (a) compares the seismogram responses at an offset of 2.5km while (b) shows the same at 7.5km offset and (c) at 10km. In (a), (b) and (c) one can observe that the VFSA seismogram is within half the dominant period of the true model response. There are a few mismatched events that are recovered after gradient FWI.

2.4.1 Field Data I: Krishna Godavari Basin, India

The Krishna-Godavari (KG) basin lies at the centre of the eastern continental margin of India with a North-East to South-West extent. The basin has several cycles of deposition starting from late Carboniferous to Holocene. However, Permian to Pliocene sands constitute the bulk of the reservoirs in this region. Pliocene sands with alternating silt sequences form the best payoff zone for gas (Sain and Gupta, 2008; Sain et al., 2012). The upward movement of gas in KG offshore leads to the formation of shallow hydrates where the pressure and temperature lie in the hydrate-stability zone (Rao, 2001). A 2D line was collected in 2010 by MV Akademik Fersman. The line was shot with 584 shots every 50 meters with 180 receivers per shot. The receivers were placed every 25 meters with the minimum and maximum offset being 100m and 4575m, respectively. To perform the inversion, I regathered 584 shots from an end-on to split-spread geometry using reciprocity (Liu et al., 2004) resulting in 360 receivers per shot in a split-spread layout and 400 shot points. The minimum and maximum offsets did not change after using this transformation. The limited offsets plus the fact that the sea bottom is nearly a kilometer deep means that I do not have any turning rays in the data and have to rely exclusively on reflections to estimate the velocity distribution using FWI.

My aim in the inversion of this hydrate-rich area was to successfully delineate the Bottom Simulating Reflector (BSR) located at the bottom of the hydrate layer. The wavelet used in the inversion process was computed

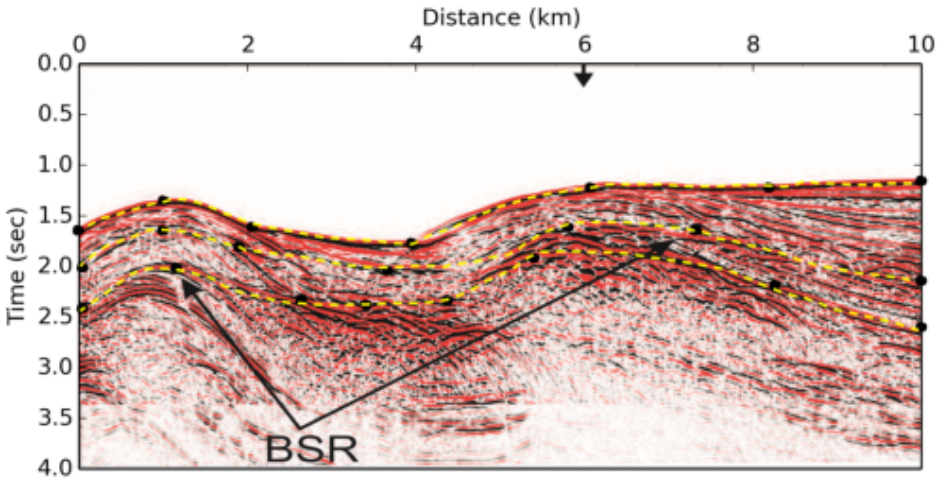


Figure 2.9: I pick the nodes for the interfaces from a near offset section of KG data. I selected nodes for three interfaces shown in black circles and the initial interpolated interface is shown by a dotted yellow lines.

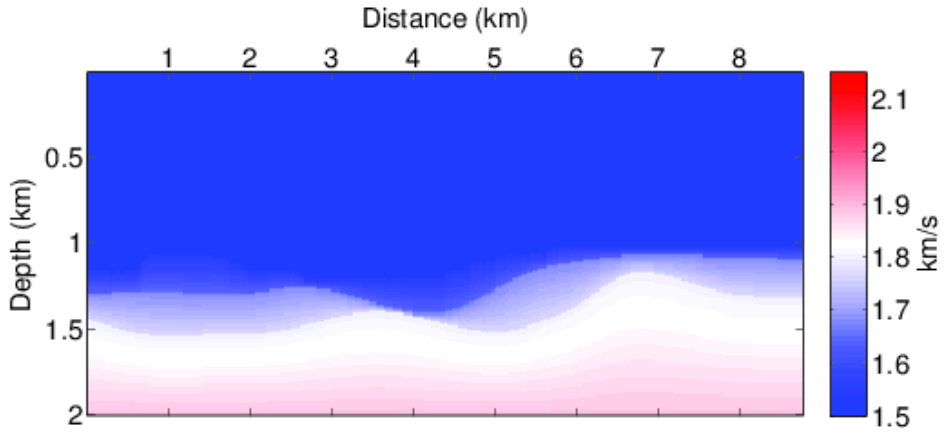


Figure 2.10: A random starting velocity model for KG data using 3 picked interfaces from Figure 2.9.

by averaging out the near-source direct wave arrivals over multiple shots. For both VFSA and gradient FWI, I filter the data to 10Hz and pick 3 interfaces from the near offset seismic section shown in Figure 2.9. I run VFSA for 200 iterations to obtain a starting model for gradient FWI. One such starting model is shown in Figure 2.10, and the model obtained after VFSA is shown in Figure 2.11. The model obtained by VFSA is smooth and does not show much velocity variation with depth. To better condition the gradient FWI I applied time damping of $e^{-0.60t}$ to the observed seismograms. This was done to better constrain the shallow part of the model with only limited offsets. I run gradient FWI for 80 iterations using a smoothed version of Figure 2.11 as the starting model. The final model after inversion is shown in Figure 2.12. The BSR is successfully delineated in the final model shown by black arrows. The convergence curves for 200 iterations of VFSA and 80 iterations of gradient show normalized error (negative of cross-correlation function) vs iteration in Figure 2.13a and Figure 2.13b respectively. The error in VFSA seems to fluctuate at the earlier iterations, which helps it escape local minima. The gradient FWI builds upon the VFSA result to converge to a model containing high frequency features. The data comparison between the observed data and the synthetic from the VFSA and FWI model is shown in Figures 2.14a, 2.14b and 2.14c, respectively. I also compare the well log, FWI model, VFSA model and smoothed version of VFSA model at $x = 6km$ shown in Figure 2.14d. The well log comparison shows that the VFSA inverted for only two interfaces rather than three as I supplied. This means that v_{up} and v_{down} came out

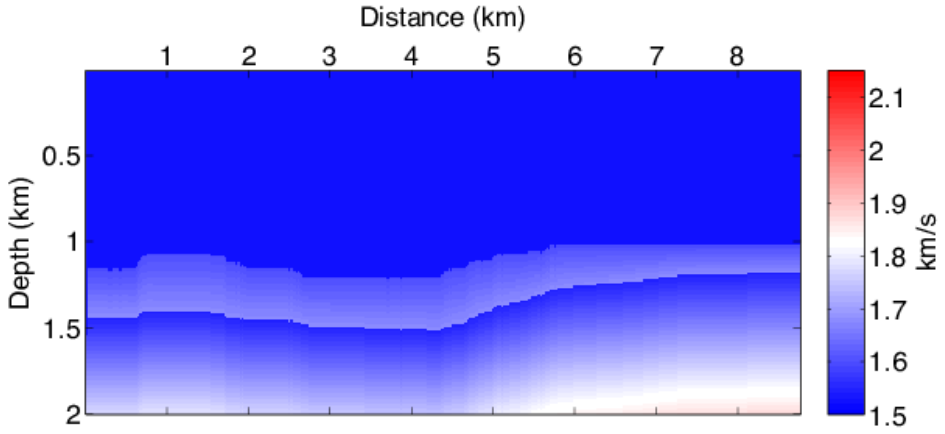


Figure 2.11: The velocity model obtained for KG data after 200 iterations of VFSA using 3 picked interfaces from Figure 2.9. The estimated velocity model after VFSA contains just two interfaces. This velocity model is smoothed and used as the starting model for a gradient-based FWI.

very close to each other. Because the velocities are extremely close, my VFSA formulation regards it as one continuous interface. Another observation is that there is a difference of about 200m/s in the velocity estimate compared to the well log and the VFSA/FWI model. I think this is because of the density variations just beneath the seafloor that were not captured by a constant density acoustic wave equation. Because one interface was collapsed by VFSA and velocity in the shallow part is under determined, the BSR predicted by the VFSA model is at a greater depth compared to the well log. This incorrect depth as well as the other interface is captured by FWI, which corrects the depth of the BSR (about 1.3km) as well as recovers the missing interface from the VFSA model.

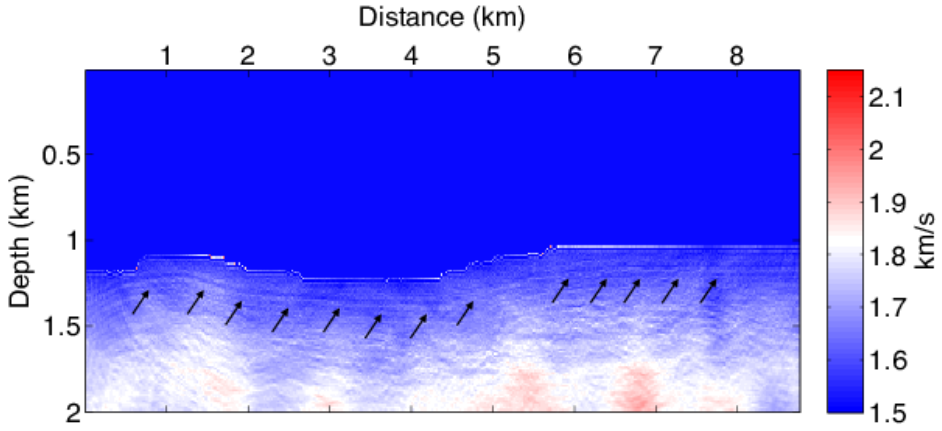
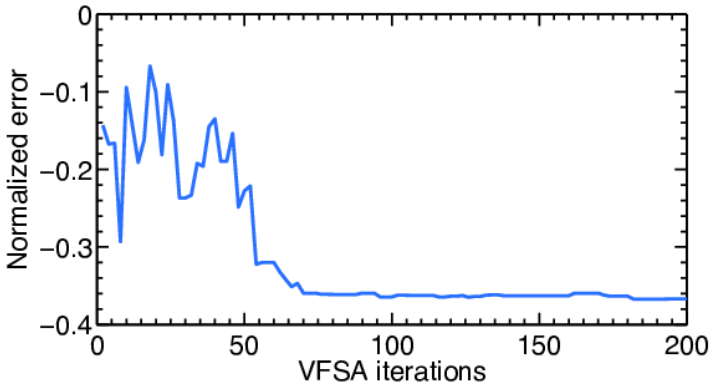


Figure 2.12: The velocity model obtained after 80 iterations of gradient-based FWI. This velocity model is able to delineate the BSR at the correct depth shown by black arrows.

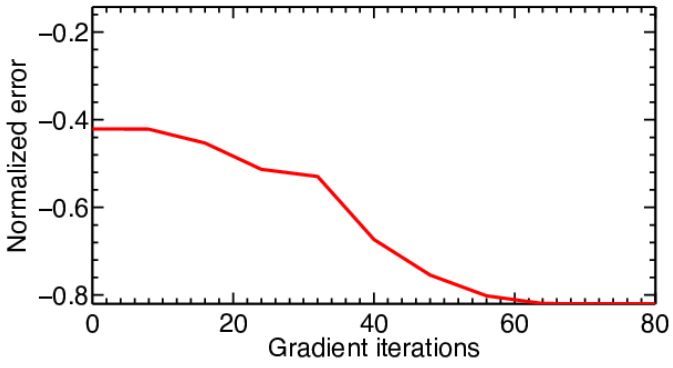
2.4.2 Field Data II: Mahanadi Basin, India

The Mahanadi basin lies at the north of India's eastern continental margin containing depositional sequence from late cretaceous to Pliocene era. The reservoirs are Pliocene sandstones capped by Paleogene shales (Subrahmanyam et al., 2008). A 2D line was shot in 2010 by MV Akademik Fersman. The line contained 1120 shots at 25 m interval and each shot had 360 receivers. The receiver spacing was 25m and the minimum and maximum offset were 100m and 9075m, respectively.

Because this dataset has considerably higher longer than the KG dataset, I was able to use the extended offsets and turning rays to better constrain the deeper part of the model. Like the KG data, I filtered the data using a 10 Hz low pass filter to perform VFSA inversion. For the gradient FWI, I used a 15



a



b

Figure 2.13: Convergence curves for the KG dataset: (a) Normalized error vs iteration for VFSA iterations to obtain a sparse starting model for FWI. (b) Normalized error vs iteration for 80 iterations of FWI with the starting model from VFSA based minimization.

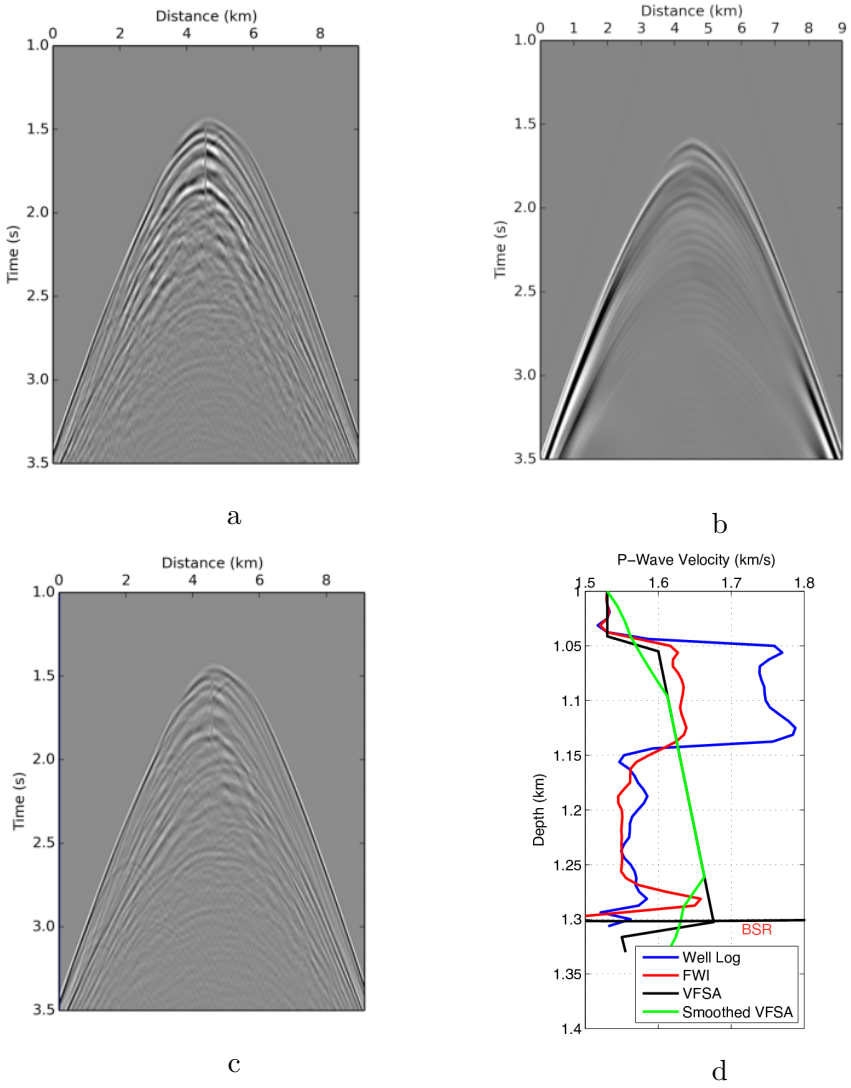


Figure 2.14: (a) shows a shot record from the KG dataset at x=2km while (b) shows the seismic response for the FWI model at the same location. They show very good agreement. (c) shows the comparison between the well log (blue) the FWI model (red) the VFSA model (black) and the smoothed VFSA model (green).

Hz low pass filter on the seismograms with a time damping of $e^{-0.25t}$ on the observed seismograms to constrain my gradient FWI. Direct waves were muted from the data and were not used in the modeled seismograms to calculate the misfit. I selected five interfaces from the near trace stack section shown in Figure 2.15 and optimized the interfaces and the velocities using VFSA showing a random starting model in Figure 2.16. The estimated velocity model after 200 iterations is shown in Figure 2.17. This model was smoothed and used as a starting model for gradient-FWI, which after 80 iterations converged to the model shown in Figure 2.18. This final model contains high wavenumber features, which are not present in the velocity model from VFSA. To confirm the accuracy of the velocity model, I overlay the velocity model over the RTM image of the data in Figure 2.19. The overlay shows that the seismic events are consistent with the velocity variations in the inverted velocity model. The convergence curves can be seen in Figure 2.20a for 200 iterations of VFSA and in Figure 2.20b for 80 iterations of gradient FWI, which behave similarly to the KG dataset. To show that FWI improves upon the velocity model estimated by VFSA, I plot the Common Image Gathers (CIGs) from the VFSA model in Figure 2.21a and that from the FWI model in Figure 2.21b. The CIGs show that the one obtained after FWI is noticeably flatter and has little residual moveouts compared to the VFSA model.

To confirm if gradient FWI builds up on the spatial features of the VFSA model, I show a comparison of the vertical wavenumber spectrum of the VFSA model (red) and FWI model (blue) in Figure 2.22. One can clearly

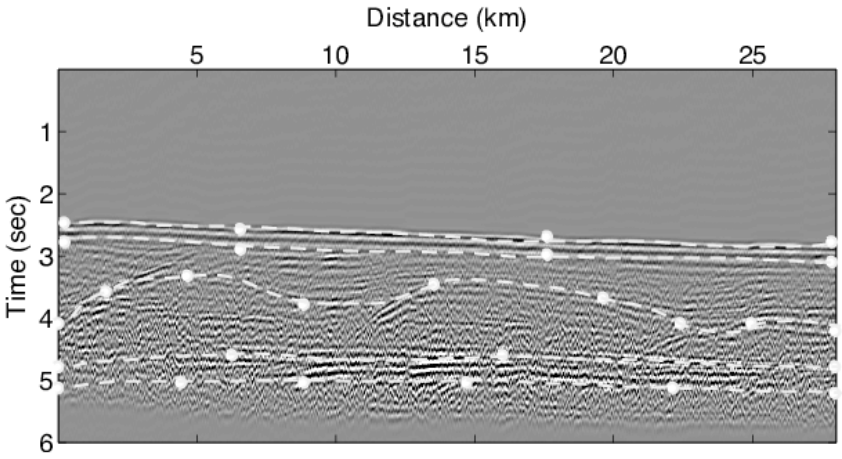


Figure 2.15: I pick the nodes for the interfaces from a near offset section of Mahanadi data where selected nodes for five interfaces shown in white circles and the events leading to the picks have been shown by dotted white lines.

observe that the FWI model fills up the voids near the low and mid wavenumber components of the spectra, thereby filling out the missing information not recoverable by VFSA using a sparse parameterization.

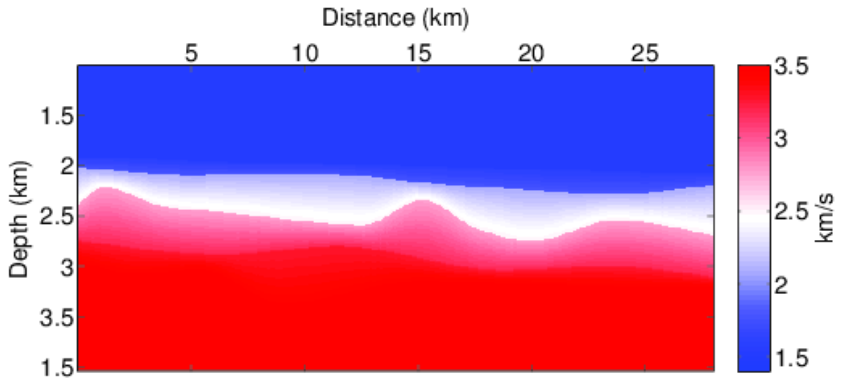


Figure 2.16: A random starting velocity model for Mahanadi data using 5 picked interfaces from Figure 2.15.

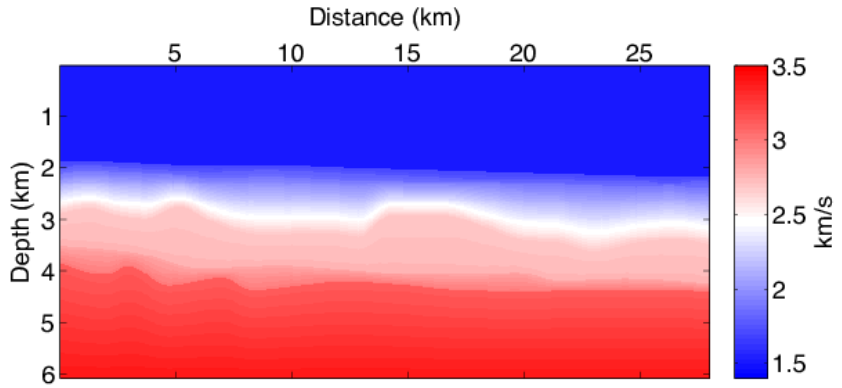


Figure 2.17: The velocity model obtained for Mahanadi data after 200 iterations of VFSA using 5 picked interfaces from Figure 2.15. The estimated velocity model contains the optimal location of interfaces. This velocity model, after smoothing, is used as the starting model for a gradient-based FWI.

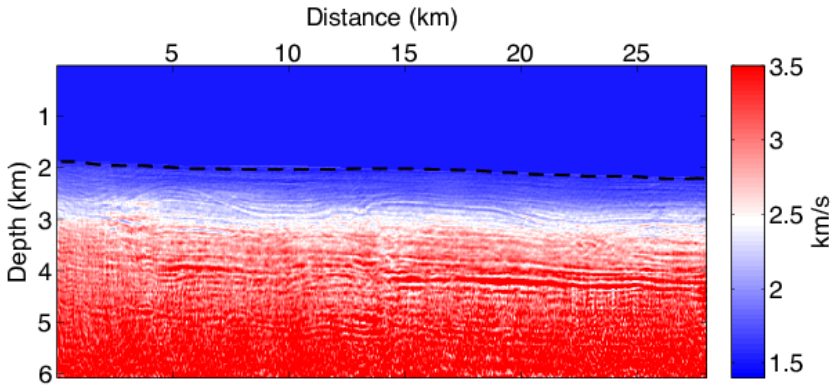


Figure 2.18: The velocity model obtained after 80 iterations of gradient-based FWI using the model from Figure 2.17 as the starting model. The seafloor is shown by the dotted black line and the model contains all high wavenumber features not contained in the VFSA model.

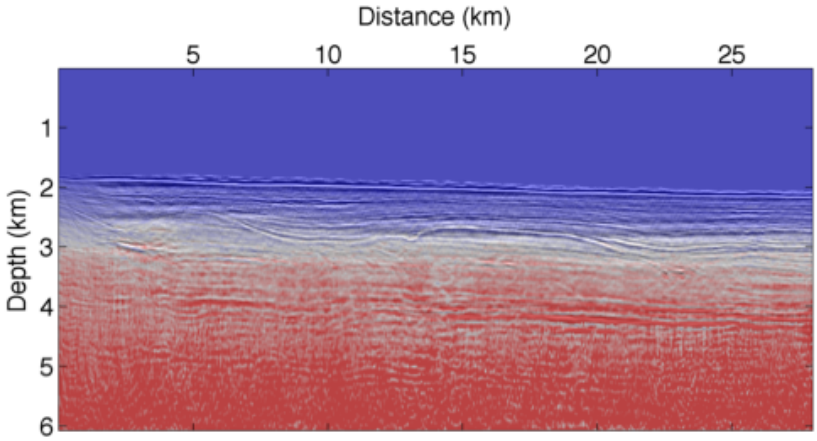


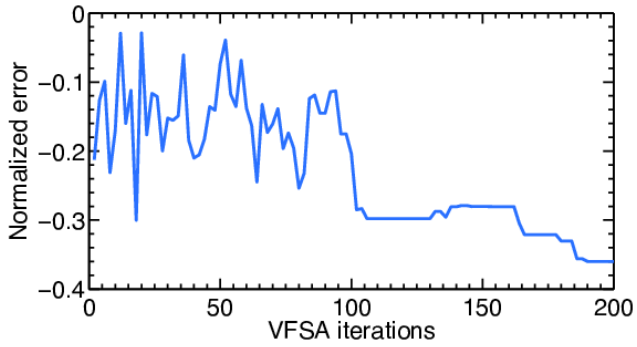
Figure 2.19: An overlaid image of the velocity model obtained after FWI for the Mahanadi data over its corresponding RTM image. The events in the RTM section are well correlated with velocity changes.

2.5 Discussion

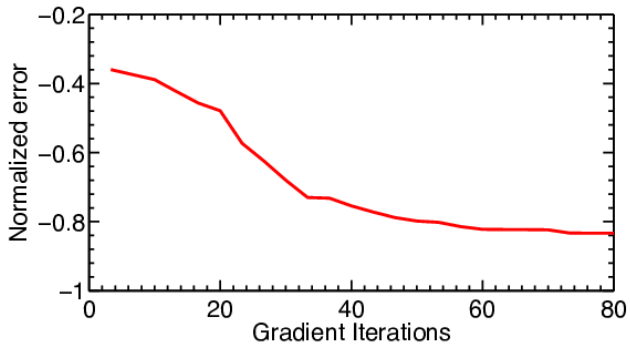
In the preceding sections, I demonstrated that a two-step inversion of VFSA followed by gradient FWI is able to produce acceptable data-fit and generate realistic velocity models. Global Optimization methods such as Simulated Annealing are known to be computationally expensive. However, my implementation of VFSA with sparse parameterization converges fairly quickly making it practical for FWI. Note that my objective in VFSA inversion is not to fit the data completely (my sparse parameterization does not allow that) but to model major events in the seismograms. In the following subsections, I address some of the critical aspects of VFSA.

2.5.1 Sensitivity towards the starting model and effect of priors

I perform different tests to establish the stability of the VFSA inversion under different and inaccurate starting guesses. Figure 2.23 shows the search space of position of the nodes for the synthetic case where only the first and the third interface are shown for clarity. The red boxes form the limits of the search space, and I run 10 realizations of VFSA using different starting guesses randomly drawn from the search space for each node represented by blue dots. The final inverted interface from each realization is plotted with a dotted black line. It is easy to see that while the starting guess spans a fairly large variation in the search space, the final inverted interfaces exhibit little to no variation proving that my VFSA approach is insensitive to the starting guess under acceptable bounds. Another interpretation of this result is that

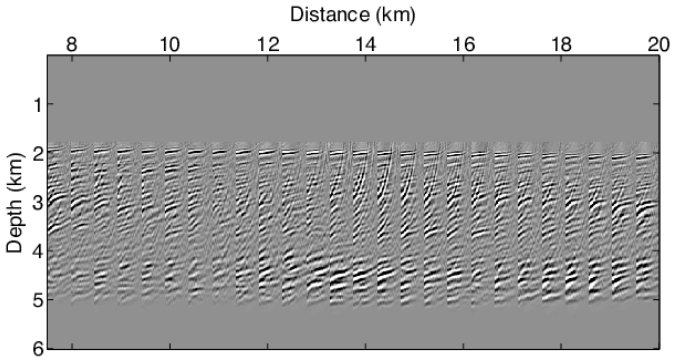


a

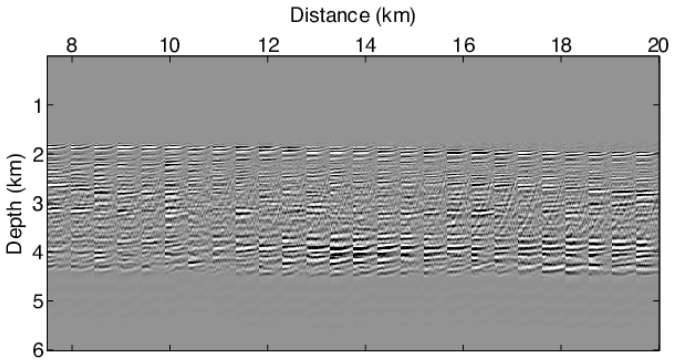


b

Figure 2.20: Convergence curves for the Mahanadi dataset: (a) Normalized error vs iteration for VFSA iterations to obtain a sparse starting model for FWI. (b) Normalized error vs iteration for 80 iterations of FWI with the starting model from VFSA based minimization.



a



b

Figure 2.21: Comparison of Common Image Gathers(CIGs) for the Mahanadi data (a) shows the CIGs from the VFSA model and (b) shows the CIGs from the model after FWI. CIGs from the VFSA model (a) are markedly flatter than the VFSA model (a)

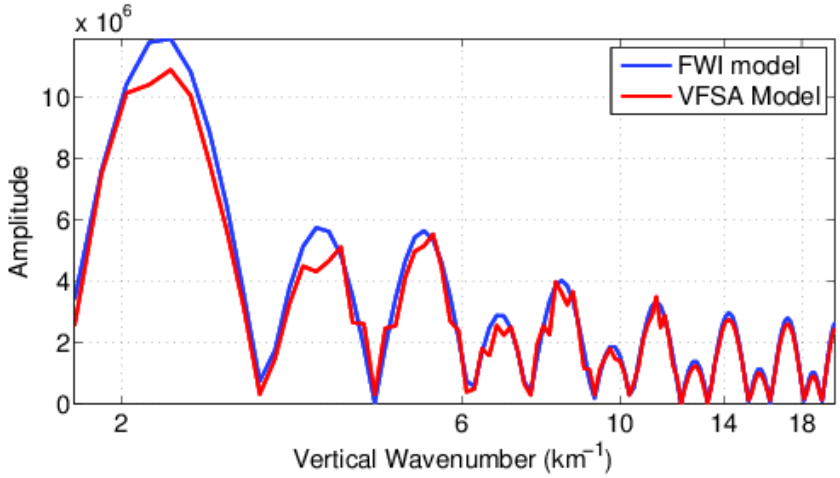


Figure 2.22: Comparison of the vertical wavenumbers contained the final model after FWI (blue) and the model after VFSA (red). One can see the gradient FWI (blue) filling up the missing wavenumbers from the VFSA model (red). This hints at the fact that the VFSA builds the low wavenumber component of the model and gradient FWI fills up mid to high wavenumbers thereby reconstructing an accurate velocity model.

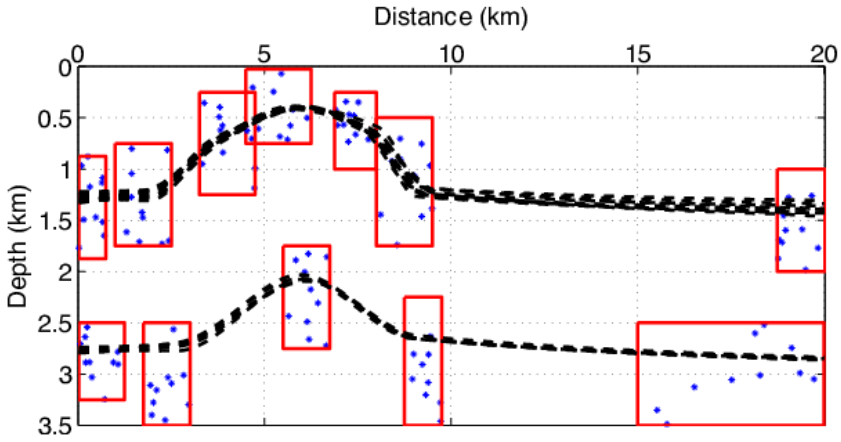


Figure 2.23: Figure showing the stability of the inverted model to different initial guesses. I show 2 two interfaces from the synthetic test for clarity where the search window for each node is given by red rectangles and run 10 realizations of VFSA using different starting models. The starting guesses are shown in blue dots while the final interfaces obtained after each realization is shown with dotted black lines. The inverted interface appears to be invariant to the search space converging to nearly the same points with different starting guesses. The 4 velocities obtained in this inversion are also withing 5 percent of each other.

the picking of interfaces may not be rigorous provided a fairly generous search space is used in the VFSA inversion.

2.5.2 Choice of VFSA parameters

I ran my VFSA inversion algorithm for 200 iterations using a starting temperature of 500 and a cooling schedule given by $T = T_0 0.98^n$, where n is the iteration number and 0.98 is the temperature decay factor. I explain my reasoning to select 200 iterations based on an inversion test performed using a different number of iterations and decay factors. Figure 2.24 shows three

VFSA runs on synthetic data for 400 iterations using different decay constants 0.98 (blue), 0.8 (green) and 0.99 (red). It is easy to observe that cooling too fast (decay=0.8) causes premature convergence (green) where the algorithm is unable to explore the model space in an optimal way. Convergence happens in about 100 iterations but the final result is one of the local minima obtained as a result of cooling too fast. A slow decay factor of 0.99 (red) leads to a slower convergence at around 300 iterations after which the normalized error does not go down any further. An optimum decay factor of 0.98 (blue) facilitates faster convergence than the red curve at around 150 iterations with the final model exhibiting the same error as the red curve. This decay factor has the optimal balance of model exploration and exploitation giving us the benefits of cheaper costs and reasonably accurate convergence in fewer iterations, which forms my motivation to select 200 iterations of VFSA with a decay factor of 0.98. However, it should be mentioned that an analysis like the one shown above should be done for different data because depending on the complexity of the data, the parameters most likely will change. I am not claiming that 200 iterations with 0.98 decay works for every application, but it will change depending on the complexity of the problem. However, with some experience, one should be able to choose optimal parameters more and more easily.

2.5.3 Computational overhead from VFSA inversion

Another aspect of VFSA that needs some explanation is the added computation cost of VFSA inversion. I run 200 iterations of VFSA, which is

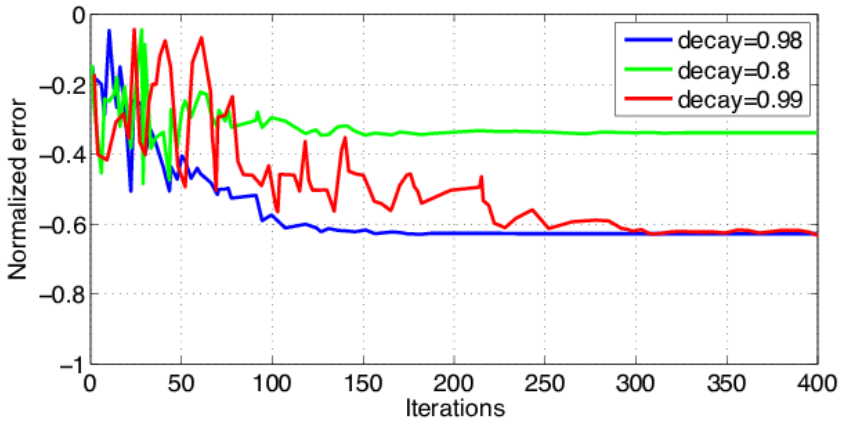


Figure 2.24: Convergence behavior using different decay factors. The red, blue and green curves show the convergence behavior using decay factors of 0.99, 0.98 and 0.80, respectively. Using a steep decay factor of 0.8 (green) leads to premature convergence to a non-optimal model. Using a slow cooling schedule of 0.99 (red) gives an optimal model but takes many more iterations to converge. The optimal cooling schedule of 0.98 (blue) converges to the same model as (red) in fewer iterations making it my choice of cooling schedule.

substantially more than the gradient iterations. Note that number of forward model evaluations required in one iteration of VFSA and gradient FWI is not the same. VFSA is a derivative-free method, and, therefore, does not require computation of the gradient of the objective function, which saves one forward computation. It avoids storage and access of snapshots required by gradient-FWI. It does not require computation of step length either. Therefore, each iteration of VFSA requires only one forward calculation while that of gradient FWI requires the equivalent of at least five (two for gradient and three for step length) forward model evaluations. As a result, the computational cost of five iterations of VFSA is equivalent to one gradient iteration. In my tests where I run 200 iterations of VFSA, it is equivalent to 40 iterations of gradient FWI making the VFSA cost just half of the gradient cost. Furthermore, I could further reduce the computational complexity of the VFSA part by performing the inversion in the plane wave domain (Vigh and Starr, 2008) as opposed to the shot domain. Plane wave transformation creates fewer virtual gathers requiring fewer forward calculations compared to the shot domain.

2.5.4 Extension to 3D

While the idea of VFSA and FWI are both expandable to three dimensions, I am not aware of any direct formulation for non-oscillatory splines that can interpolate in three dimensions. However, Kruger (2003) presents the idea of non-constrained splines as bypassing the requirement for equal second order derivatives at each point and replacing them with their first order equivalents.

Therefore, this approach can be easily extended to three dimensions. The problem essentially remains the same of estimating a starting model using a sparse parameterized velocity model and then using gradient FWI to improve upon the starting model. It is easy to notice that an equivalent 3D problem would need to optimize many more nodes compared to a 2D problem. A direct implication of this is that VFSA would require more iterations to converge. Other than this, the VFSA framework remains the same and so does the idea of gradient FWI, which needs to be solved in 3D. I must again mention that each iteration of VFSA would be much less expensive than FWI, so the total overhead created by VFSA should not be prohibitive.

2.5.5 Multi-VFSA for uncertainty estimation

In this chapter, I obtained convergence in VFSA using 200 iterations. As discussed earlier, the number of iterations might not be the same for all problems and might increase or decrease depending on the complexity of the dataset and the number of interfaces to invert for. If the required number of iterations is low, multiple runs of VFSA (Jackson et al., 2004) is surely a good way to assess uncertainty in the macro-model estimation. One can use parameters like different starting models, shifting interfaces and even different values of starting interfaces to see how VFSA behaves with changing parameters. It is computationally tractable to perform uncertainty analysis in 2D, but doing the same in 3D, though possible, might be prohibitively expensive to do on a routine basis.

2.5.6 Other global optimization methods

Population based global optimization methods such as Genetic Algorithm Sambridge and Drijkoningen (1992), Particle Swarm Optimization (PSO) Kennedy (2010), and Differential Evolution (Saraswat et al., 2010) can also be used instead of VFSA as used in this paper. We demonstrate the population based PSO method in Appendix A for use in FWI. Although the method worked fairly well, population based methods are hard to tune for optimal performance. The computational cost is also higher compared to VFSA.

2.6 Conclusions

I demonstrated a two-step technique where the initial model for FWI containing low frequencies is estimated using a stochastic global optimization method. Using a global optimization method decreases sensitivity to the starting model and is less likely to be trapped in local minima. The initial model obtained by this method contains major interfaces and is in agreement with major arrivals in the real seismic sections. After the initial model has been estimated using VFSA, I smooth the model in the vertical direction and use it as the starting model for conventional gradient-based algorithm to recover all the higher frequency components of the velocity model. I applied the technique to one synthetic dataset and two real datasets where the proposed algorithm was successfully able to estimate the correct velocities in the final model as seen in the velocity model in the synthetic case and from the logs and images in the real data case. I also discussed the subjects of how to choose parameters

in VFSA, its stability to starting model and inaccurate priors. The computational overhead from the VFSA method is not substantial and can be easily supplemented by routine FWI. The results are encouraging, and I believe that VFSA can be easily integrated as a part of routine FWI workflow to estimate macro-models.

Chapter 3

Salt shape estimation using shape parameterization and Full Waveform Inversion

3.1 Summary

A good starting model is imperative in Full Waveform Inversion (FWI) because it solves a least squares inversion problem using a local gradient-based optimization method. A sub-optimal starting model can result in cycle skipping leading to poor convergence and incorrect estimation of subsurface properties. This problem is especially important for salt models because strong velocity contrasts create substantial time shifts in the modeled seismogram. Incorrect estimation of salt shape leads to velocity inaccuracies in the sediments because the least-squares gradient aims to reduce travel time differences without considering the different scales of velocities in them. In this paper, I propose a technique to build velocity models containing salt bodies using a combination of global and local optimization techniques as well as separating the velocity updates using a partitioned level set surface. I use Very Fast

Parts of the chapter were first published in *Datta, D., Sen, M., Liu, F., & Morton, S. (2016). Salt model building by shape-based parameterization and global FWI. In SEG Technical Program Expanded Abstracts 2016 (pp. 1069-1073). Society of Exploration Geophysicists. The work was done by Debanjan Datta under a project funded by Hess Corporation.* The full version of the paper has been submitted to the Geophysics Journal and is currently under review.

Simulated Annealing (VFSA) along with sparse parameterization to estimate a starting model. The sediments are represented by a set of interfaces and velocities while the salt body is represented by a set of ellipses. The VFSA inverted model is then used as a starting model in FWI where the sediments are updated using the least squares gradient while the salt shape is updated using a level set formulation. I demonstrate my approach using three 2D synthetic salt models namely the SEG-EAGE salt model, the Sigsbee 2A model and a modified SEG-SEAM Phase I model. A brief discussion on the robustness of the VFSA inversion and a few sources of uncertainty in the estimation of salt shapes are discussed.

3.2 Introduction

Full Waveform Inversion (Tarantola, 1984; Virieux and Operto, 2009) (FWI) is an iterative local solver that solves an optimization problem minimizing the misfit between recorded and modeled seismic data. It utilizes the entire information from the seismic waveform to compute the gradient of the misfit function using a computationally efficient method known as the Adjoint State Method after Plessix (2006). It has become the de-facto standard for velocity estimation problems in academia and industry alike.

Despite the popularity of FWI, it suffers by design due to the local nature of the optimization method. A sub-optimal starting model far from the global minimum can cause cycle skipping and make it converge to an inaccurate model with higher misfit and mismatched velocities. It is, there-

fore, essential to obtain an optimal starting model before running the FWI workflow. Typical pre-processing steps to estimate the starting model include first-arrival traveltime tomography (Nolet, 2012), stereotomography (Billette and Lambaré, 1998; Lambaré, 2008) or migration velocity analysis(Sava and Biondi, 2004a,b). These methods require human intervention to work well; they are often effectively expensive operations because they are carried out multiple times before a good model can be estimated.

Several techniques have been proposed over the years in order to avoid the issue of cycle skipping without needing much user input during the process. The Multiscale Method (Bunks et al., 1995) is one of the examples implemented to reduce premature convergence by widening the global minimum trough using stepped frequencies when building the model. A popular approach to tackle cycle skipping is using modified objective functions that are able to compensate for large time shifts between the recorded and modeled seismograms. Some examples include Laplace Domain FWI (Shin and Min, 2006), Tomographic FWI (Almomin et al., 2012) and Auxilary Bump functional FWI (Bharadwaj et al., 2016). Methods based on deconvolution and signal processing include Adaptive FWI (Warner and Guasch, 2014) and Dynamic Image Warping (Ma and Hale, 2013) that are capable of introducing large velocity updates to fix large time shifts between the observed and modeled data.

Recently, global optimization methods (Sen and Stoffa, 2013) have started to gain popularity in FWI (Aleardi and Mazzotti, 2017; Datta and

Sen, 2016; Sajeva et al., 2016). These methods are less likely to be trapped in local minima. At the same time, they converge slowly especially when the number of model parameters is high. An alternative approach to making global optimization viable in a FWI workflow is to use some kind of sparse parameterization to reduce the number of model parameters required for inversion. Sajeva et al. (2016) used genetic algorithm to estimate the velocity model using a coarse grid representation; Datta and Sen (2016) used a combination of velocities and interface nodes to estimate the velocity models with very few parameters.

Here, I expand the work of Datta and Sen (2016) to include the estimation of complex salt bodies using a sparse parameterization and VFSA framework. It is difficult for the interface-velocity formulation to represent salt bodies as they span closed shapes and have huge velocity contrasts across them. In theory, a salt body can be represented by multiple constrained interfaces, but in most cases, one does not have sufficient prior information to enforce such a constraint in the inversion method. Constraining interfaces without sufficient prior information introduces uncertainty in salt estimation because of the sharp sediment-salt interface. Local FWI with salt models depends on the closeness of the true salt shape because the higher salt velocity can introduce large time shifts in the seismogram, which can lead to cycle skipping.

I present an extension of the velocity-interface method to estimate complicated salt bodies. To do this, I combine the velocity-interface parameter-

ization technique (Sen and Frazer, 1985; Zelt, 1997) with an approach that allows sparse representation of any 2D shape. The velocity model is separated into sediments and salt body. The sediments are parameterized by the velocity-interface combination, and the salt body is parameterized by a set of ellipses, which is an optimal approach to represent any 2D shape as shown by Panagiotakis and Argyros (2016). Each ellipse is parameterized by five parameters, namely x and z co-ordinates of the center, major and minor axes and the angle of the major axis with respect to the horizontal. I estimate a macro-model with an optimal set of velocities, interfaces, and ellipse sets.

The resulting model has a smooth representation of the sediments from the true model as well as a salt shape that is fairly close to the true salt body. In order to estimate the true sediment velocities and salt shape, it is required to further update the sediment velocities and the salt shape. However, for the standard FWI workflow, it is difficult to update the salt shape because of different scales between the sediment and the salt. Therefore, I separate the sediment velocity update where I use the adjoint state gradient, and the salt body is updated by modifying a level set surface (Sethian, 1996).

Level-set methods help to naturally differentiate the sediment velocity and the salt boundary updates. The sediment is represented in a gridded form while the evolving salt boundary is represented by zero crossings of a level set surface. The sediments are updated by scaled gradient obtained from the adjoint state method while the salt is updated by updating a level set surface and taking its zero crossing. Level set methods for salt boundary updates

have been implemented in both 2D (Dahlke et al., 2015; Kadu et al., 2017; Lewis et al., 2012) and 3D (Lewis and Vigh, 2016). It has also been used in combination with radial basis functions with estimation of simple background velocities (Kadu et al., 2017). The resulting model has a better estimate of the salt shape and sediment velocity compared to the initial model.

I present the theory of my approach in the next section which, includes describing the VFSA formulation as well as the partitioned level set formulation. Then I apply the proposed method to three synthetic models with varying degrees of complexity in the salt body. In the final section of the chapter, I discuss some sources of uncertainty in this approach and the strategies to minimize them.

3.3 Theory

3.3.1 Full Waveform Inversion

The reader is referred to section 1.2 for an overview on FWI. I use the frequency domain formulation in this chapter.

3.3.2 Very Fast Simulated Annealing

Very Fast Simulated Annealing (VFSA) (Ingber and Rosen, 1992; Sen and Stoffa, 2013) is a global optimization algorithm inspired from the physical process of annealing. It does not require a good starting model to converge to the global minimum and is ideally able to start from any random position in the search space.

VFSA, like the physical process of annealing, aims to minimize the energy (error) of a system. It is defined by a dimensionless parameter called temperature, which at every iteration, dictates the perturbation and acceptance of the perturbed model. I start with a high temperature evaluating a random model and perturb it using a temperature-dependent Cauchy distribution given by

$$\mathbf{m}_{new} = \mathbf{m}_{old} + \mathbf{y}(\mathbf{m}_{max} - \mathbf{m}_{min}), \tag{3.1}$$

where $\mathbf{m}_{old}$ is the old model, $\mathbf{m}_{new}$ is the new model and y_i is given by

$$y_i = sgn(u - 0.5)T[(1 + T)^{2u-1} - 1)], \tag{3.2}$$

where T is the temperature, sgn is the signum function and u is a random number in $(0, 1)$. The resulting model is evaluated using equation 1.3 and depending on the error E, it is accepted or rejected by computing a probability given by

$$P = e^{\frac{-\Delta E}{T}}, \tag{3.3}$$

where P is the probability, ΔE is the difference in error between the current and perturbed model and T is the temperature. The perturbed model is accepted if

$$P > rand(), \tag{3.4}$$

where $rand()$ is a random number in $(0, 1)$ (Metropolis et al., 1953). The probability given by this criterion gradually diminishes by decreasing temperature,

meaning that the acceptance of worse solutions reduces by decreasing temperature. This allows VFSA to perform model exploration at higher temperatures and gradually switches to model exploitation as the temperature reduces. The steps are repeated at each iteration with lowering temperatures to obtain an optimum model. The reader is referred to Sen and Stoffa (2013) for a detailed description of VFSA.

3.3.3 Sparse Parameterization of sediments: Velocity Interface Approach

Datta and Sen (2016) proposed a sparse parameterization technique where they described a macro-model using a few interfaces and two velocities across each interface. The interfaces in the macro-model are represented by a few points (x, z), which are interpolated using non-oscillatory splines (Kruger, 2003). Once the interfaces have been interpolated, the velocity field between the interfaces is estimated by linear interpolation in the vertical direction. This parameterization reduces the number of model parameters by several orders of magnitude compared to a gridded model, while still being able to represent macro velocity models accurately. The goal of this approach is to estimate an optimal set of interfaces and velocities, which minimizes the data misfit in the sparse domain.

3.3.4 Sparse parameterization of salt bodies using ellipses

In order to reduce the number of model parameters, it is important that one uses a sparse parameterization to represent random 2D shapes. Panagio-

takis and Argyros (2016) presented a method that allows to represent any 2D shape with a set of ellipses. The main idea behind this representation is to maximize the shape coverage of the ellipse sets compared to the true shape and to minimize the number of ellipses required for maximum coverage. Most of the shapes can be represented fairly well by 10 or fewer ellipses. Each ellipse has 5 unique unknowns namely x and the z coordinate of the center of the ellipse, the major and minor axes and the angle the major axis makes with the horizontal axis. This reduces the number of model parameters greatly as one can represent a 2D shape using just a few parameters.

Because one continually updates the ellipse parameters, it can lead to disconnected ellipse sets or a shape with jagged edges, which are undesirable in a sparsely parameterized salt body. Therefore, to create smooth edges, I regularize the ellipse sets by taking the restricted concave hull (Edelsbrunner et al., 1983) using Delaunay triangulation (Lee and Schachter, 1980). The concave hull computes the envelope of a set of points using a convex or non-convex polygon. The smoothness and convexity of the restricted concave hull depends on an auxiliary term α_{hull} where decreasing values of α_{hull} will produce bounding hulls with increasing convexity. A $\alpha_{hull} = 0$ generates the convex hull (Asaeedi et al., 2017). This effect can be seen in Figure 3.1 where the salt body estimate has a representation that is not indicative of the true salt body. After regularization with a restricted concave hull, the resulting salt body shown in Figure 3.2 is much smoother and represents a salt body almost like the true one. This, of course, leads to quicker convergence of the VFSA

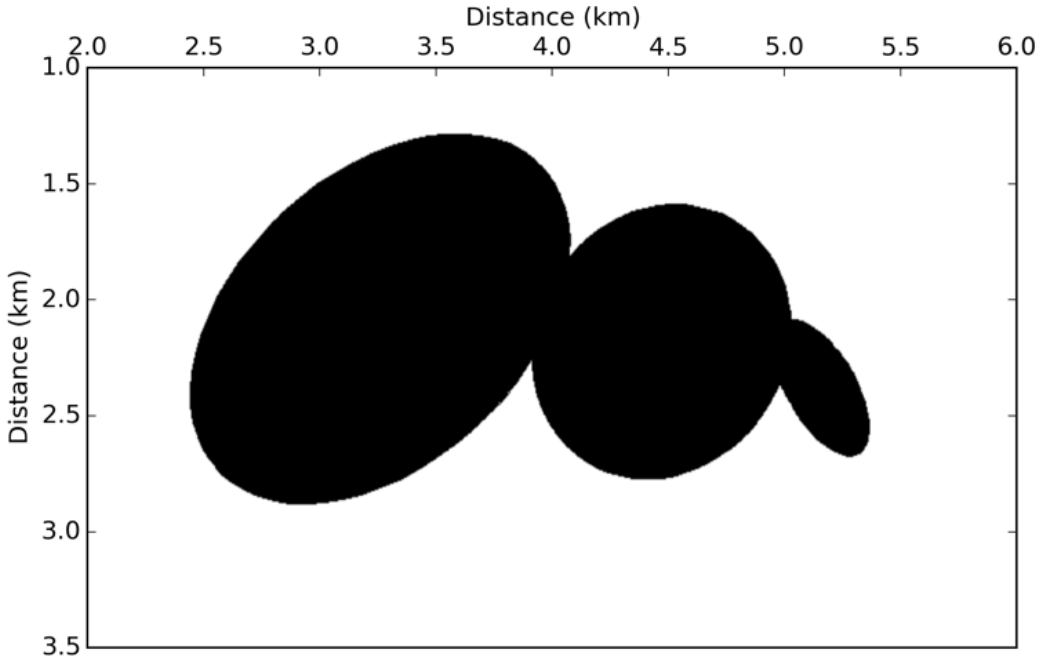


Figure 3.1: An ellipse-set representation of the SEG-EAGE salt model. The Ellipses in this example are barely touching each other leading to ragged edges which are not representative of a salt body in a sparse domain.

algorithm. Random realizations of velocity-interface and the ellipse sets are combined to compute full velocity models.

3.3.5 FWI using Adjoint state gradient

The salt body and sediment velocity obtained from the VFSA method are not completely accurate but close to the true model. To obtain the correct sediment velocities, it is essential that the salt shape is accurate because inaccurate salt shape can lead to spurious velocity updates in the sediment. However, updating the salt body using a standard gradient update approach

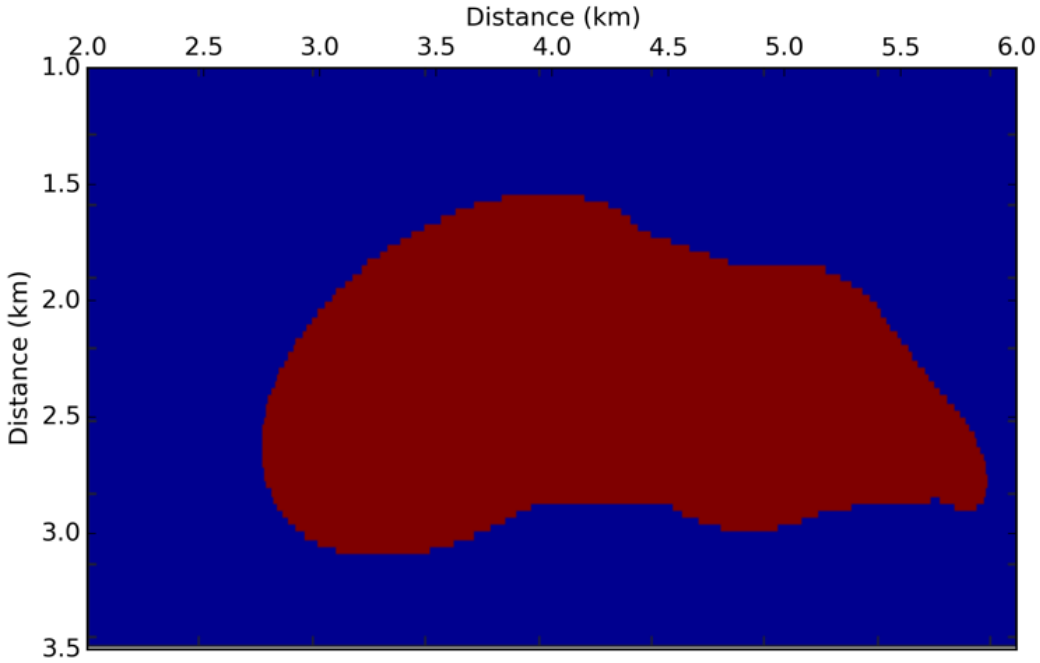


Figure 3.2: The ellipse set representation of SEG-EAGE salt model after taking the restricted concave hull of the ellipse sets. The edges are much smoother and better represent the true salt body.

is very difficult. I implement a way to update the salt boundaries independent of sediment updates using level set methods (Sethian, 1996).

Level-set methods allow us to independently vary the shape of the salt without worrying about the interference of model updates between the sediments and salt. I use three different strategies to update different parts of the model. The sediments away from the salt are updated by the standard steepest descent method using the gradient from adjoint state method. The area close to the salt body is updated by an interpolation method like Natural Neighbour Interpolation (Sambridge et al., 1995), which extends the velocity values of the sediments near the salt boundary. This is done so that the spurious gradient values near the incorrect salt boundaries do not pollute the sediment updates near the salt body.

The salt boundary is represented by the zero crossing of a binary valued surface that is positive inside the salt and negative outside. The level set function is given by

$$\phi = -1 \textbf{ outside salt body}, \tag{3.5}$$

and

$$\phi = 1 \textbf{ inside salt body}. \tag{3.6}$$

The sharp jump at the salt-sediment boundary of the level set surface is smoothed using a signed distance function (Li et al., 2010). To update the

salt boundary, I compute the gradient of the level set surface from the adjoint state gradient given by equation 1.4.

$$\frac{\partial \phi}{\partial \tau} = \frac{\partial E}{\partial v(\mathbf{x})} \frac{\nabla \phi}{|\nabla \phi|}, \tag{3.7}$$

where τ refers to the boundary of the level-set surface (Dahlke et al., 2015). The level set surface is then updated using

$$\phi_{i+1} = \phi_i + \alpha \frac{\partial \phi}{\partial \tau} + \lambda R_{LSF}, \tag{3.8}$$

where ϕ_i is the level set surface at iteration i. R_{LSF} is the regularization term (Li et al., 2010) that prevents the level set surface from developing jagged edges by forcing a signed distance function on level set evaluation. α and λ are the step lengths for the level-set gradient and the regularization term, respectively. The newer salt boundary is the zero crossing of the new surface ϕ_{i+1}.

In all my tests, I fix the top of the salt model and assume that the salt velocity is known. This better constrains the VFSA and leads to faster convergence.

3.4 Numerical Results

I apply the proposed method to three synthetic salt models.

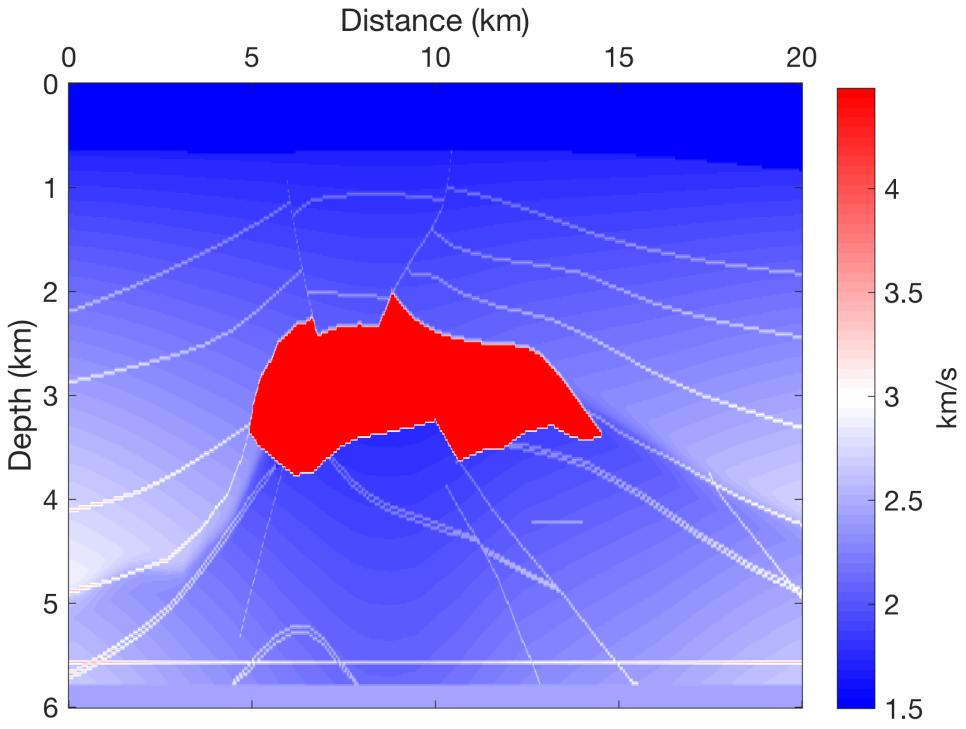


Figure 3.3: The true SEG-EAGE velocity model. The model spans 1349 points in the horizontal direction and 419 points in the vertical direction with a 15m spacing.

3.4.1 SEG-EAGE salt model

The simplest model in my tests is the SEG-EAGE salt model (Aminzadeh et al., 1997a) shown in Figure 3.3. The model contains 419 grid points in the z direction and 1349 points in the x direction with a grid spacing of 15m. I generate 100 shot gathers at a shot spacing of 60m with a receiver spacing of 15m and a total of 1064 receivers. For the VFSA inversion, I generate 2 mono-frequency datasets at 3 and 4 Hz, respectively.

I select 3 interfaces and 3 ellipses to sparsely parameterize the model. The total number of unknowns is 67 and a fairly wide search space on the ellipse sets and the interfaces were used to allow model exploration. To display the search space of the ellipses, I show the grid points that represent the salt body in all realizations of VFSA in Figure 3.5. I ran my VFSA algorithm for 200 iterations with 5 runs per temperature. The final model is shown in Figure 3.4. The error plot of the VFSA algorithm showing model exploration in the earlier iterations and exploitation in the later iterations is shown in Figure 3.6. The salt shape in the estimated model is fairly close to the true salt shape. Figure 3.7 compares the VFSA estimated salt body to the true one. The red and blue colors represent over and under parameterized salt regions respectively. It can be clearly seen that the salt mismatch is very small compared to the size of the salt body, and even the estimated sediment velocity is a good approximation to the true velocity. To demonstrate that the VFSA result does not suffer from cycle skipping, I compare the seismograms between the true and the VFSA seismogram in Figure 3.8. The seismograms appear to be very similar without any cycle skipping. For more clarity, I plot a single trace with an offset of $6km$ from the two seismograms, in Figure 3.9. The traces are well matched and can potentially converge to the true model using a local optimization method.

I run partitioned FWI with level-sets using the VFSA result as the starting model and run 60 iterations/group of partitioned level sets. The inversion is run in 3 frequency groups starting from 3 to 8 Hz. The model after 80 iterations is shown in Figure 3.10. The estimated model shows im-

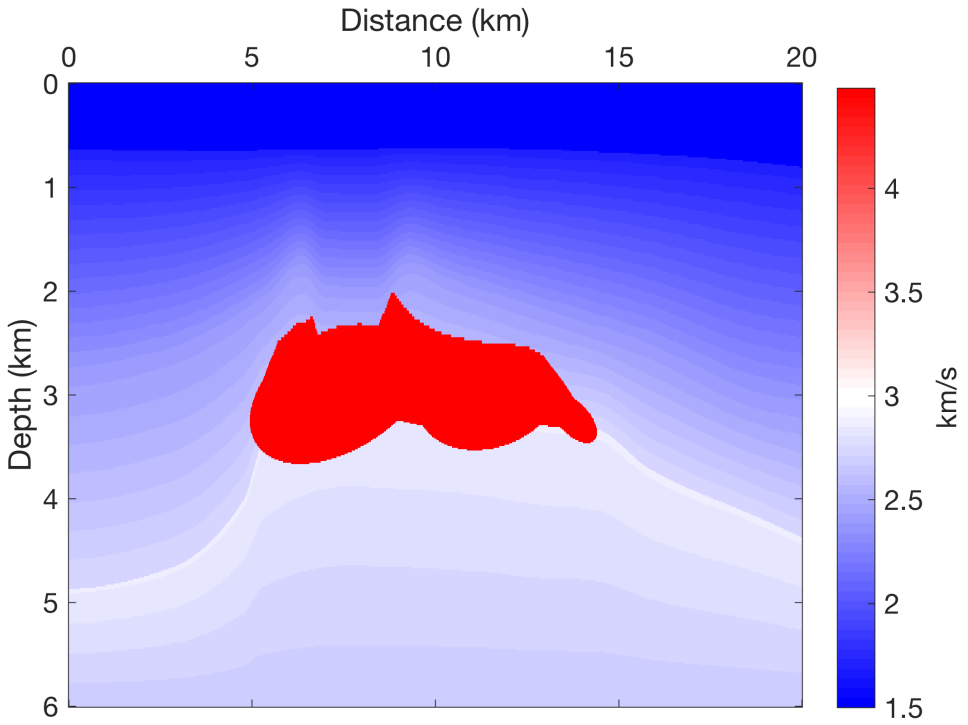


Figure 3.4: The estimated model obtained after 200 iterations of VFSA on the SEG-EAGE salt model. The model is parametrized by 4 interfaces and 3 Ellipses. The salt shape is very close to the true model, and the sediment velocity follows the low frequency trend of the sediments

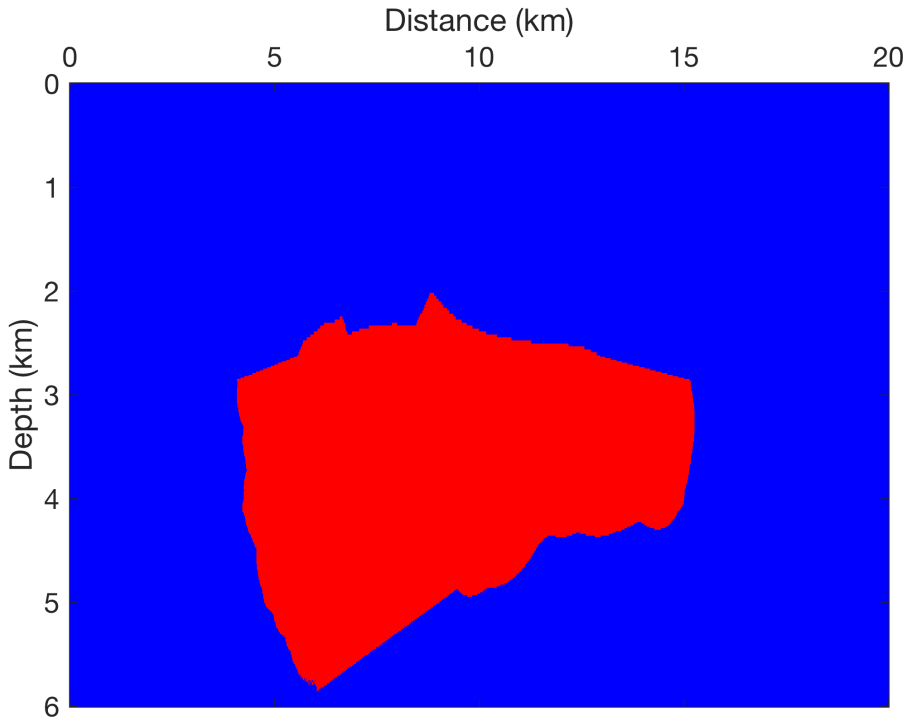


Figure 3.5: Figure visualizing the search space of the salt body during the VFSA run for the SEG-EAGE salt model. The red zone is the cumulative representation of the salt body from all sampled models in VFSA

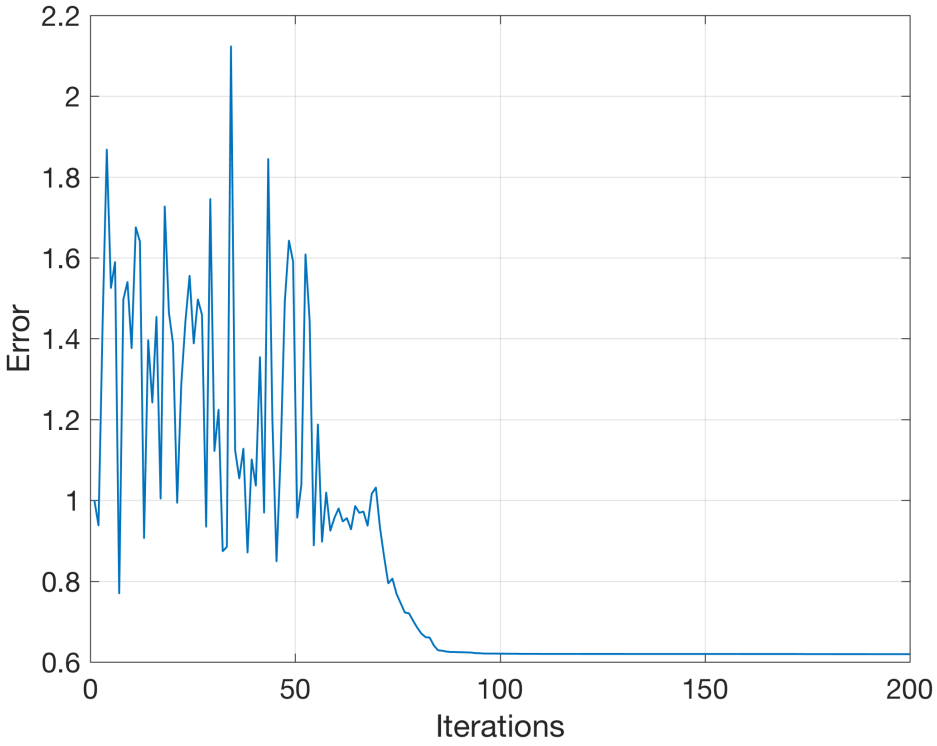


Figure 3.6: Relative error vs iteration during VFSA for the SEG-EAGE salt model. The initial iterations have oscillating error as worse models are being accepted, but the later iterations refine the model by only accepting models with a lower misfit. The acceptance to forward evaluation ratio was about 14 percent

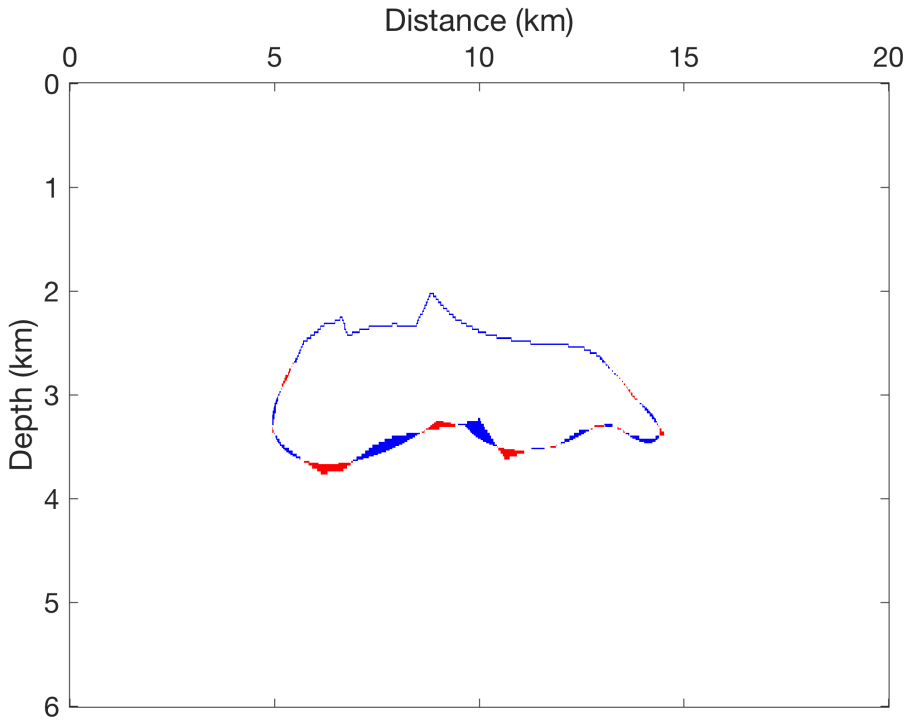


Figure 3.7: Comparison between the salt shape of the True SEG-EAGE vs VFSA estimated salt. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The VFSA estimated salt body is fairly close to the true salt body.

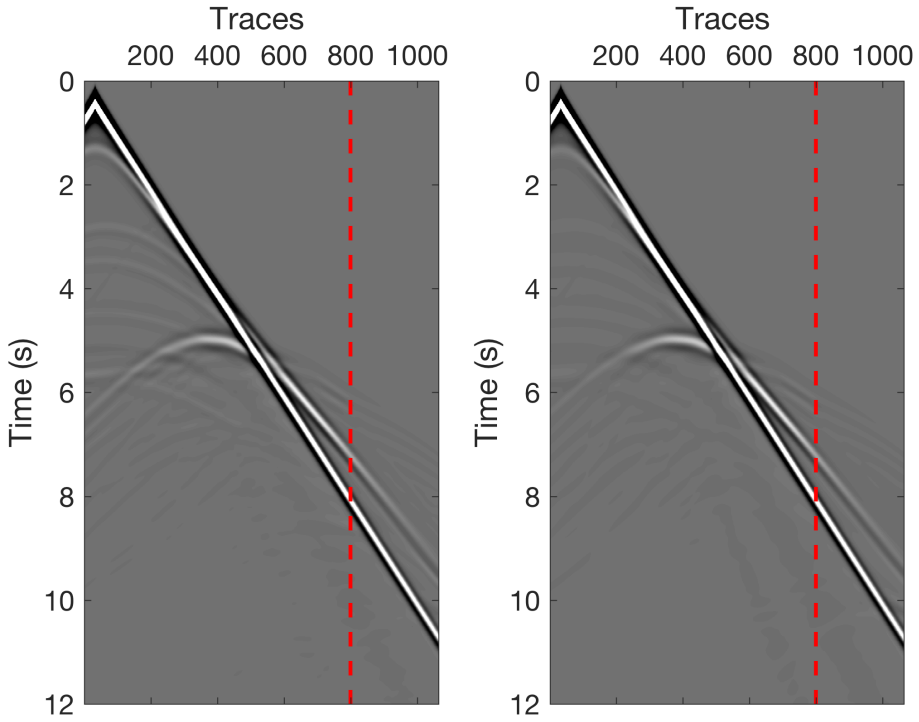


Figure 3.8: Comparison of seismogram with a central frequency of 3 Hz from the true SEG-EAGE model on the left and VFSA estimated model on the right. The seismograms appear to be free from cycle skipping.

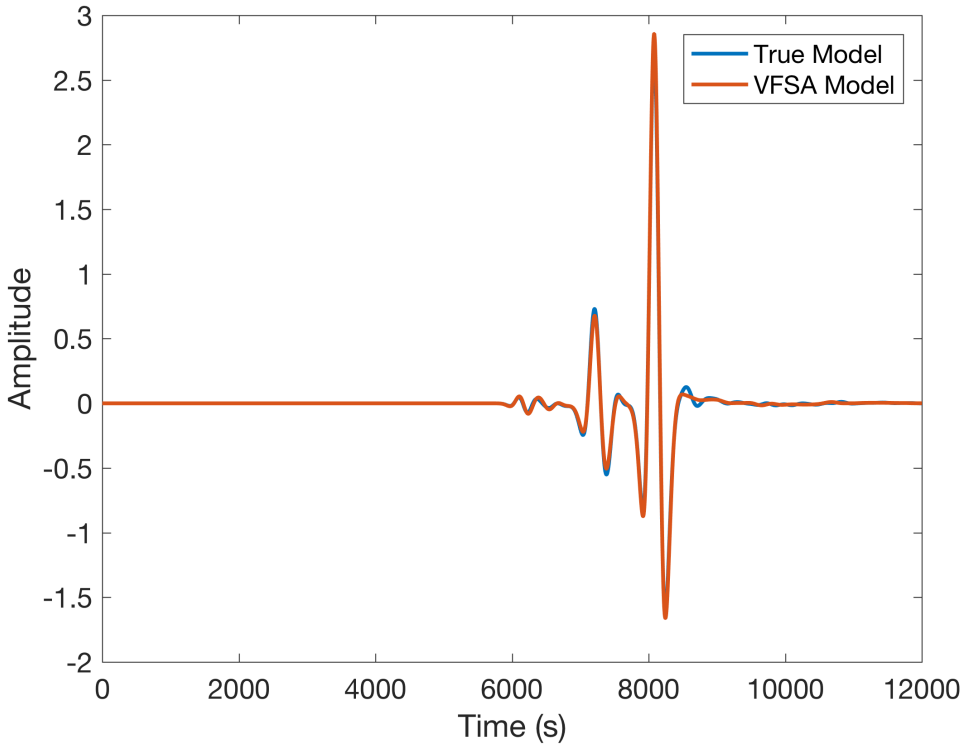


Figure 3.9: Comparison of traces at trace no 400 with the true trace shown in blue and the VFSA trace shown in red. The seismograms are pretty closely matched and show no evidence of cycle skipping.

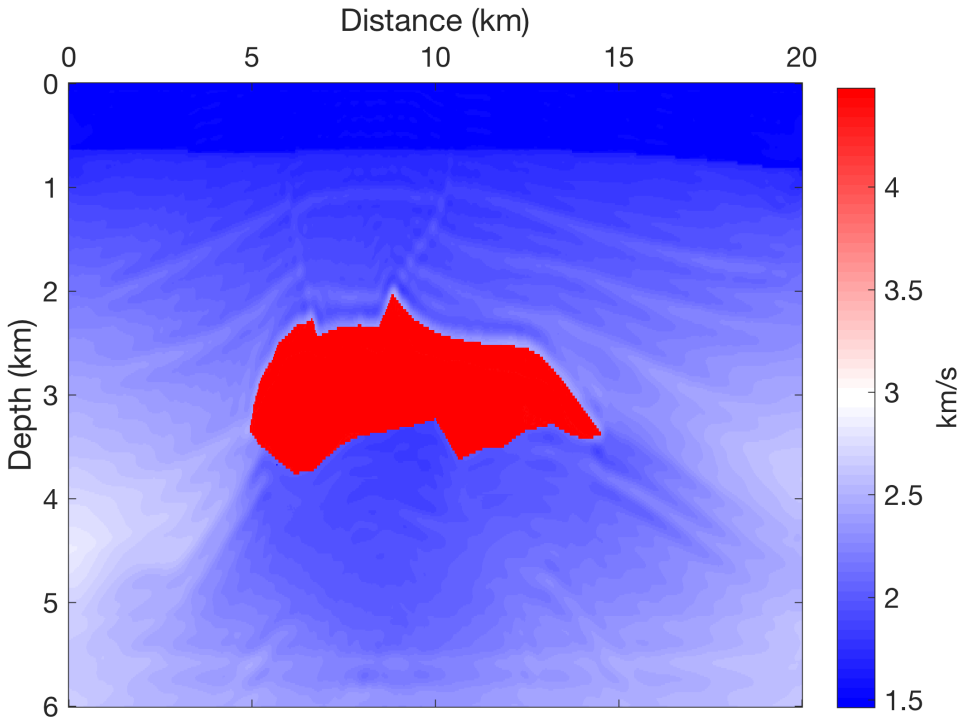


Figure 3.10: The final estimated velocity from SEG-EAGE model after 60 iterations of partitioned Level set FWI using the model from Figure 3.4 as the starting model. The salt body is better estimated after FWI and the sediments are also recovered fairly well.

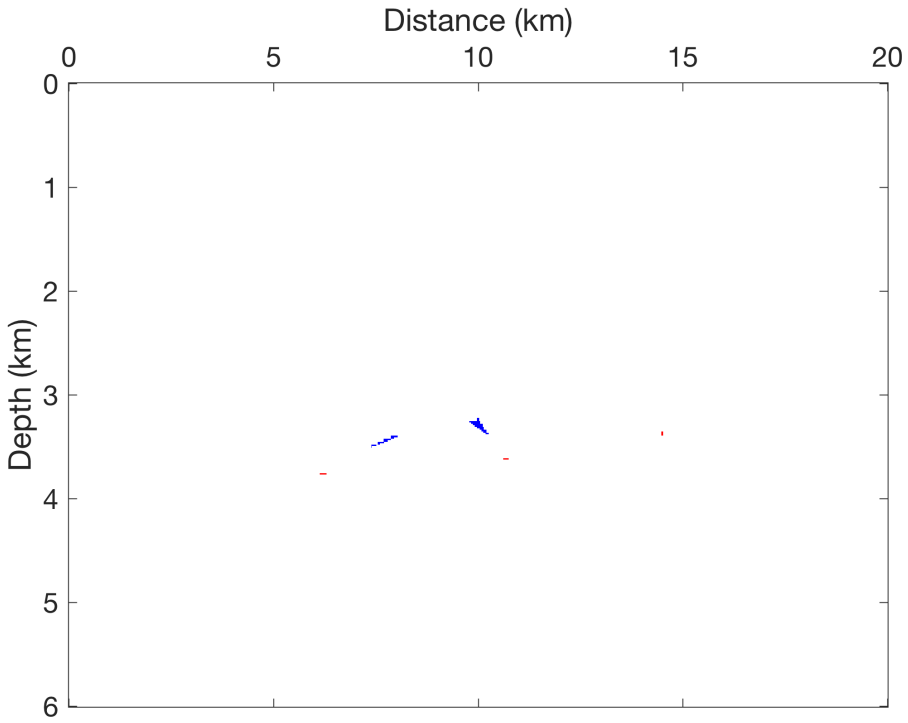


Figure 3.11: Comparison between the salt shape of the True SEG-EAGE vs FWI model. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The FWI salt body shows marked improvement over the VFSA estimate.

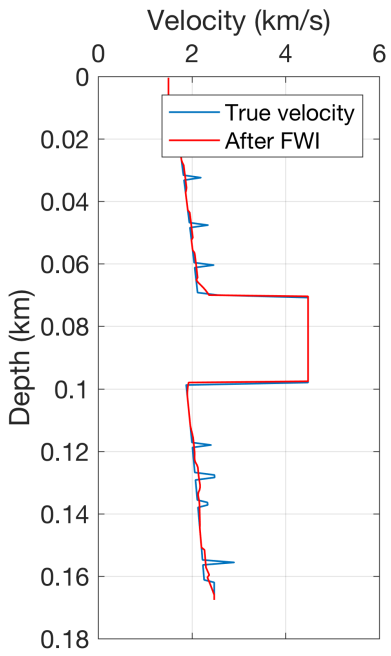


Figure 3.12: Comparison between the vertical velocity profile between the true SEG-EAGE vs FWI model.

proves initial estimates of sediment and salt bodies from the VFSA model. The comparison between true salt shape and FWI estimated salt is shown in Figure 3.11. Comparing Figure 3.7 and Figure 3.11 one can conclude that the partitioned FWI is able to improve the initial salt body. I compare the vertical velocity section of the true and FWI model at $x = 6km$ in Figure 3.12. The salt extent matches perfectly, and the sediments also show perfect match everywhere except the zones of sharp velocity jumps. This is because we used a limited frequency band to perform FWI.

3.4.2 Sigsbee 2A model

The Sigsbee2A model(Paffenholz et al., 2002) shown in Figure 3.13 is discretized with 401 points in z direction and 1065 points in the x direction, respectively, with a grid spacing of 20m. I generated 500 shots with a shot interval of 40m, and receivers were placed every 20m. Like the SEG-EAGE model, I generate seismic data at 3 and 4Hz and run VFSA inversion for 200 iterations with 5 runs per iteration. For the salt body, I use a set of 5 ellipses, and the sediments were parameterized by 3 interfaces. The search space of the salt body was fairly wide as shown in the superimposed realizations of random models during the VFSA inversion in Figure 3.14. The estimated model from VFSA is shown in Figure 3.15 and has a good estimate of both the sediments and the salt body. The comparison between the true salt body and the VFSA estimated salt, shown in Figure 3.17, demonstrates that the salt body estimate is very close to the true one. The seismograms generated

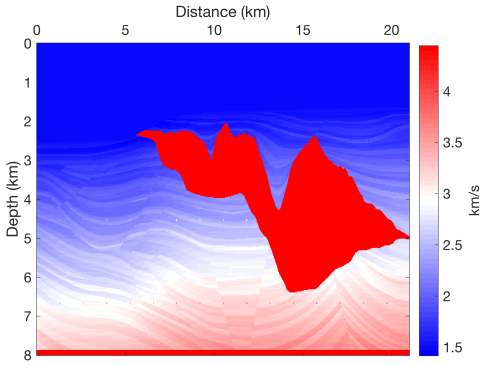


Figure 3.13: The true Sigsbee2A velocity model. The model spans 1065 points in the horizontal direction and 401 points in the vertical direction with a 20m spacing.

from the VFSA model also compares well with the seismograms from the true model without any visible cycle skipping as seen in Figure 3.18. For more clarity, I also compare a single trace for the true vs VFSA seismic response at an offset of $4km$ in Figure 3.19.

I use the VFSA estimate as a starting model in the partitioned level-set FWI for 60 iterations/group, and the final model is shown in Figure 3.20. The model estimated after FWI is able to recover the sediment velocities as well as a more accurate salt body compared to the VFSA result. The comparison between the true and the FWI salt bodies are shown in Figure 3.21. The vertical velocity profile of the true and FWI at $x = 8km$(Figure 3.22) shows similar characteristics of the previous model where the salt extent as well as the sediments above the salt match perfectly.

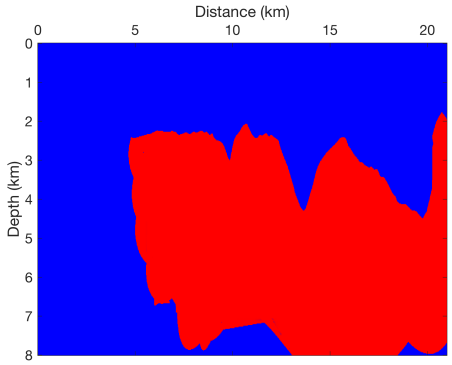


Figure 3.14: Figure visualizing the search space of the salt body during the VFSA run for the Sigsbee2A salt model. The red zone is the cumulative representation of the salt body from all sampled models in VFSA.

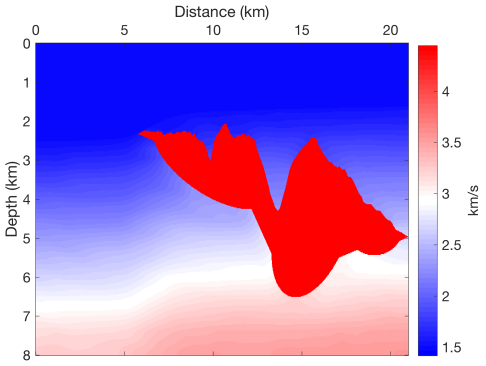


Figure 3.15: The estimated model obtained after 200 iterations of VFSA for the Sigsbee2A model. The model is parametrized by 4 interfaces and 5 Ellipses. The salt shape is very close to the true model and the sediment velocity follows the low frequency trend of the sediments.

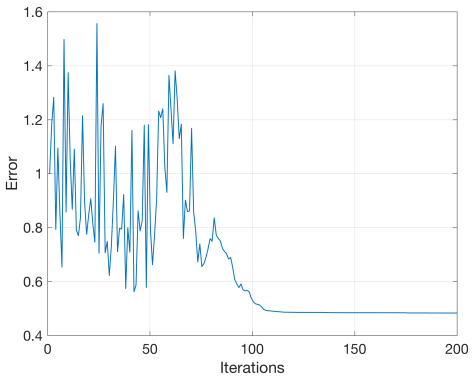


Figure 3.16: Relative error vs iteration curve in the VFSA run for the Sigsbee2A model. The initial iterations have oscillating error as worse models are being accepted but the later iterations refine the model by only accepting models with a lower misfit. The acceptance to forward evaluation ratio was about 16 percent.

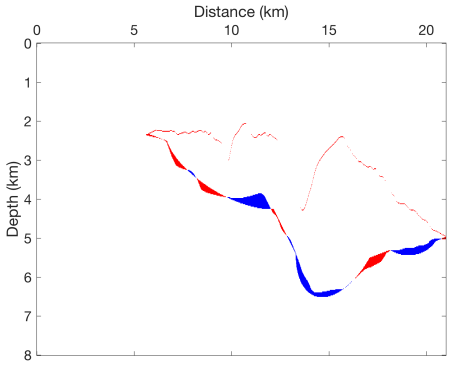


Figure 3.17: Comparison between the salt shape of the true Sigsbee2A vs VFSA estimated salt. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The VFSA estimated salt body is fairly close to the true salt body.

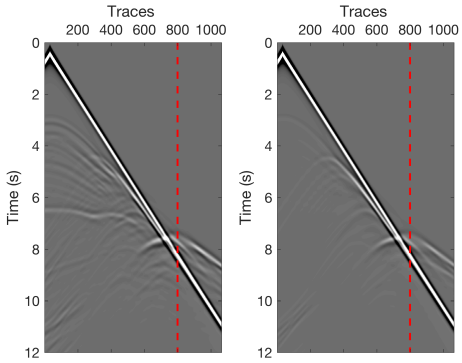


Figure 3.18: Comparison of seismogram with a central frequency of 3 Hz from the true Sigsbee2A model on the left and VFSA estimated model on the right. The seismograms appear to be free from cycle skipping.

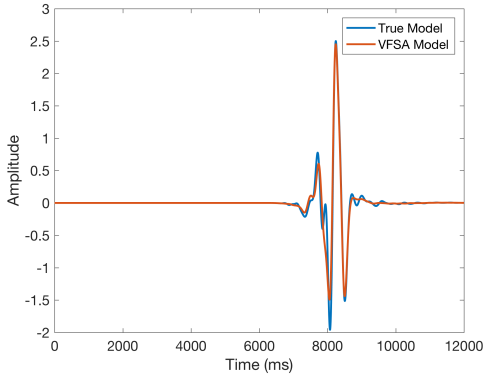


Figure 3.19: Comparison of traces at trace no 800 with the true trace shown in blue and the VFSA trace shown in red. The seismograms closely match and show no evidence of cycle skipping.

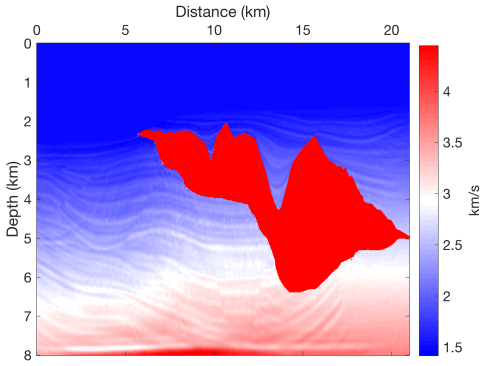


Figure 3.20: The final estimated velocity from Sigsbee2A model after 60 iterations of partitioned level-set FWI using the model from Figure 3.15 as the starting model. The salt body is better estimated after FWI, and the sediments are also recovered fairly well.

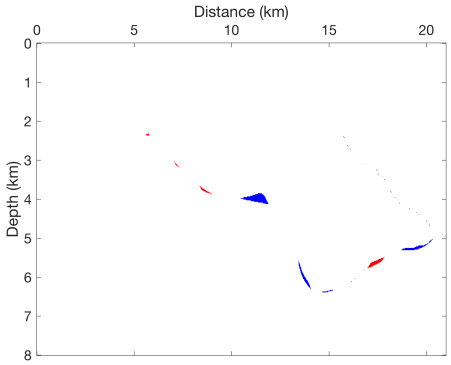


Figure 3.21: Comparison between the salt shape of the True Sigsbee2A vs FWI model. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The FWI salt body shows marked improvement over the VFSA estimate.

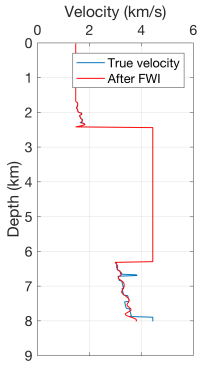


Figure 3.22: Comparison between the vertical velocity profile between the true Sigsbee2A vs FWI model.

3.4.3 Modified SEG-SEAM Phase I model

I use a modified version of the SEG-SEAM phase I model shown in Figure 3.23. The salt body is pushed deeper under the sea floor compared to the original model to avoid strong shallow reflections from the top of the salt and the dirty salt on the original model is converted into a clean salt. Also, the velocity in the deeper part of the model is modified to obtain 2 distinct salt bodies. The velocity model is 1488 points in the x direction and 848 points in the z direction with a grid spacing of 10m. I generate 92 shots with a shot interval of 160m and 1488 receivers with a receiver interval of 10m. From a heavily smoothed version of the true model (eg. like velocity analysis), I select a rough shape of the salt as shown in Figure 3.24 based on velocity values. I then use that shape to estimate 9 ellipses.

I select 4 interfaces with 8 velocities for the sediment and 9 ellipses for the 2 salt bodies. The total number of unknowns is 145. VFSA was run

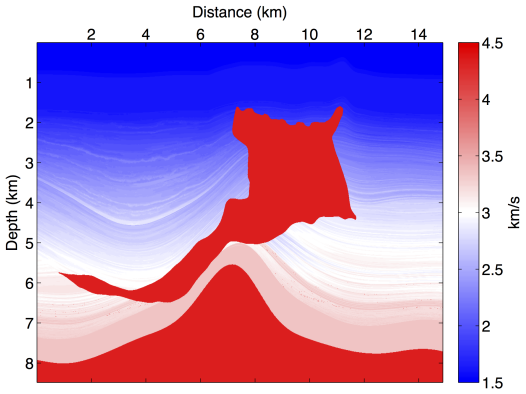


Figure 3.23: The modified SEG-SEAM velocity model. The model spans 1488 points in the horizontal direction ans 848 points in the vertical direction with a 10m spacing.

for 200 iterations with 5 moves per temperature to obtain a final model as shown in Figure 3.26. I show the search space in Figure 3.25 superimposing the realizations of salt bodies estimated during the VFSA process. The error vs iteration curve is shown in Figure 3.27 showing model exploration in the earlier iterations and model exploitation in the later iterations. The VFSA model gives a good indication of the sediment and salt bodies. The comparison between the VFSA and true salt model is shown in Figure 3.29. The comparison between the true and VFSA seismograms, shown in Figure 3.28 shows no evidence of cycle skipping.

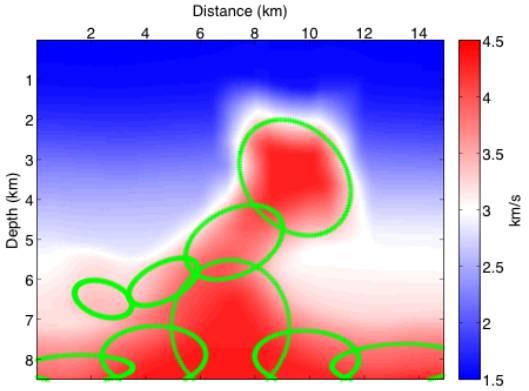


Figure 3.24: A rough shape of the salt body (v $\geq$ 3700) picked from a heavily smoothed true model. The picked shape can be represented by 9 ellipses shown by green dots.

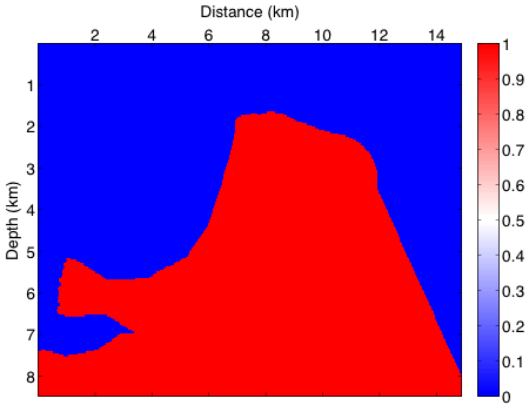


Figure 3.25: Figure visualizing the search space of the salt body during the VFSA run for the SEG-SEAM salt model. The red zone is the total representation of the salt body from all sampled models in VFSA.

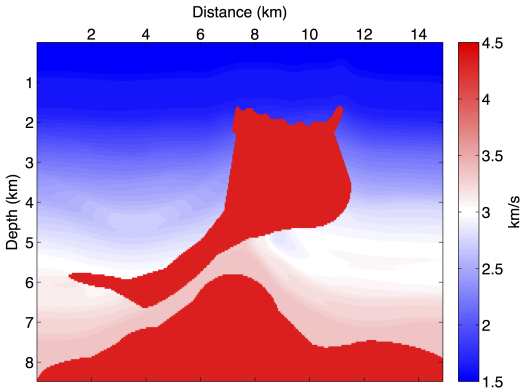


Figure 3.26: The estimated model obtained after 200 iterations of VFSA for the SEG-SEAM model. The model is parametrized by 4 interfaces and 9 Ellipses. The salt shape is very close to the true model and the sediment velocity follows the low frequency trend of the sediments.

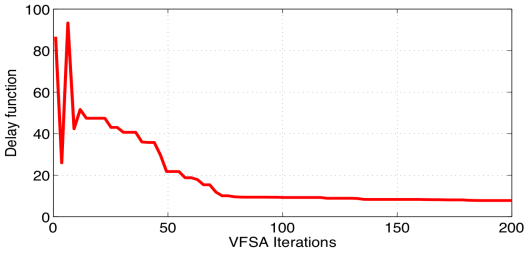


Figure 3.27: Relative error vs iteration curve in the VFSA run for the SEG-SEAM model. The initial iterations have oscillating error functions as worse models are being accepted but the later iterations refine the model by only accepting models with a lower misfit. The acceptance to forward evaluation ratio was about 6 percent.

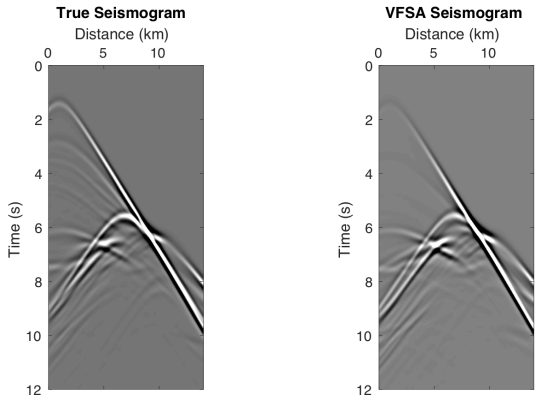


Figure 3.28: Comparison of seismogram with a central frequency of 3 Hz from the true SEG-SEAM model on the left and VFSA estimated model on the right. The seismograms appear to be free from cycle skipping.

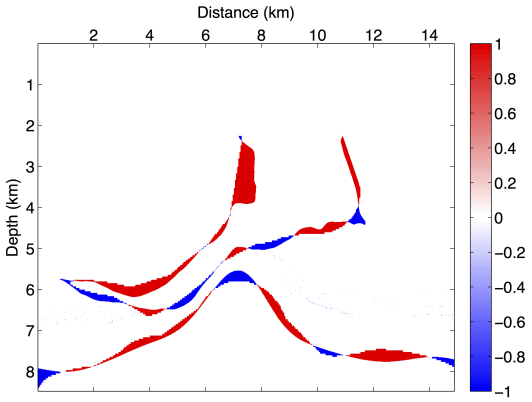


Figure 3.29: Comparison between the salt shape of the True SEG-SEAM vs VFSA estimated salt. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The VFSA estimated salt body is fairly close to the true salt body.

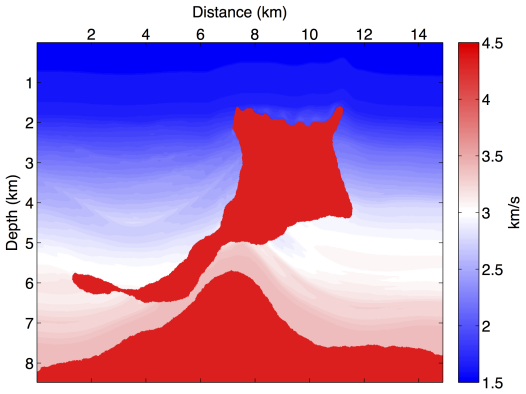


Figure 3.30: The final estimated velocity from SEG-SEAM model after 60 iterations of partitioned Level set FWI using the model from Figure 3.26 as the starting model. The salt body is better estimated after FWI and the sediments are also recovered fairly well.

I use the VFSA estimate as a starting model for partitioned FWI using 3 frequency groups ranging from 3 to 5 Hz for 60 iterations/group and the final model is shown in Figure 3.30. Compared to the final VFSA model in Figure 3.26, the salt body from the final local FWI result is much closer to that in the true model. The background velocities are also much closer to the true velocities. One observation from Figure 3.30 is that out of the two salt bodies, the top one in general shows more improvement than the bottom one. This can be attributed to the illumination differences between the salt bodies. I solved the level set evolution for both bodies together without correcting the gradient for illumination effects. One can see that the lower salt body gets smaller steps than the top one and, therefore, does not show as much change as the top one.

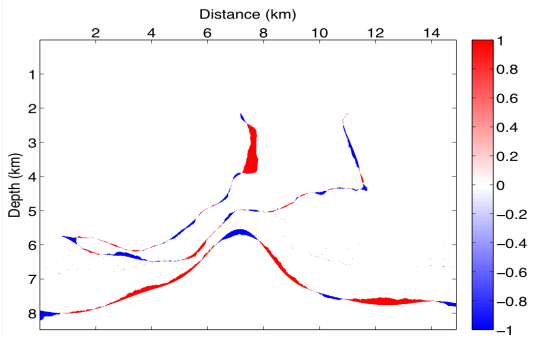


Figure 3.31: Comparison between the salt shape of the True SEG-SEAM vs FWI model. Blue denotes the area where VFSA estimated sediments in place of salt while red denotes the areas where VFSA estimated salt in place of sediments. The FWI salt body shows marked improvement over the VFSA estimate.

3.5 Discussions

We discuss a few sources of uncertainty in the VSFA results and how to tackle them

3.5.1 Uncertainty from different starting models

Because VFSA is a stochastic algorithm, different starting model from different runs can potentially lead to different final models. To quantify this uncertainty, I ran the VFSA inversion 5 times each for the SEG-EAGE salt model and the Sigsbee2A model. I plot the mutually exclusive result for all the five salt bodies and show the result in Figure 3.32 and 3.33 respectively. The areas in red show the area in the salt that are not common to all five salt bodies. The Figure shows so little mismatch between different runs that one can easily conclude that the final result is independent of the starting model.

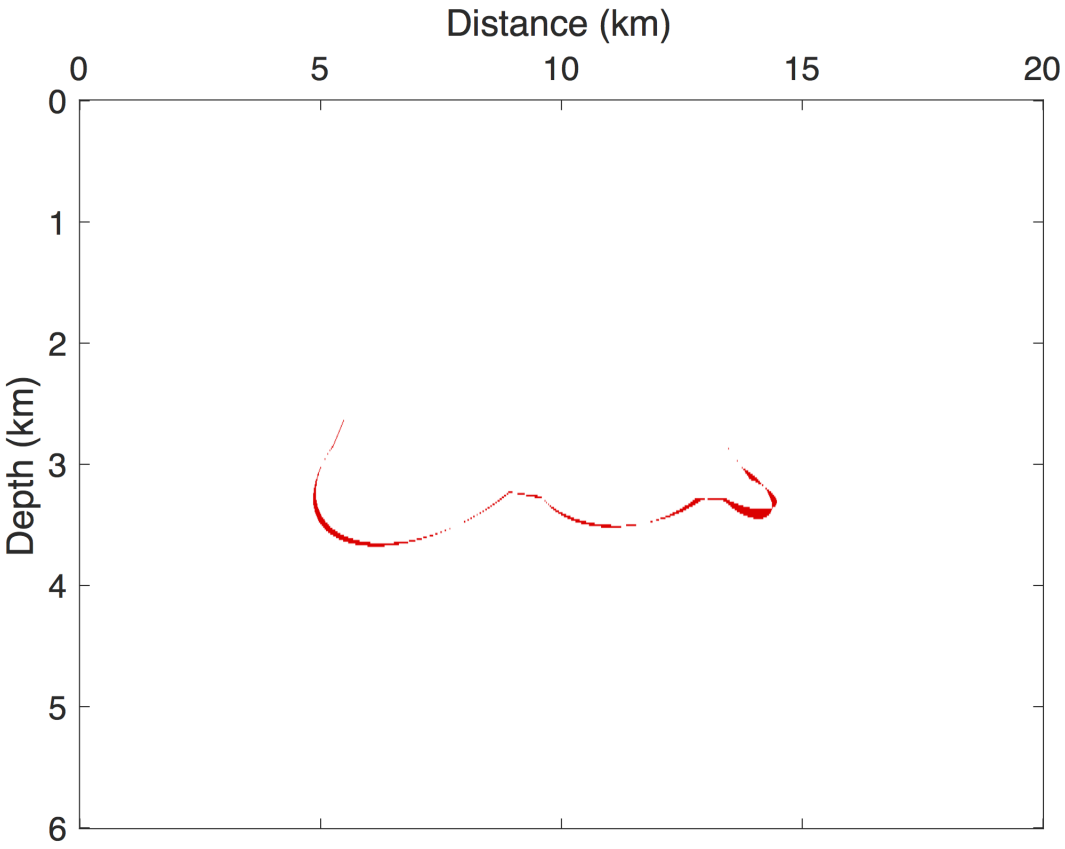


Figure 3.32: The red part shows the part of the salt not covered by all five realizations of VFSA run for the SEG-EAGE model. The uncertainty is very small.

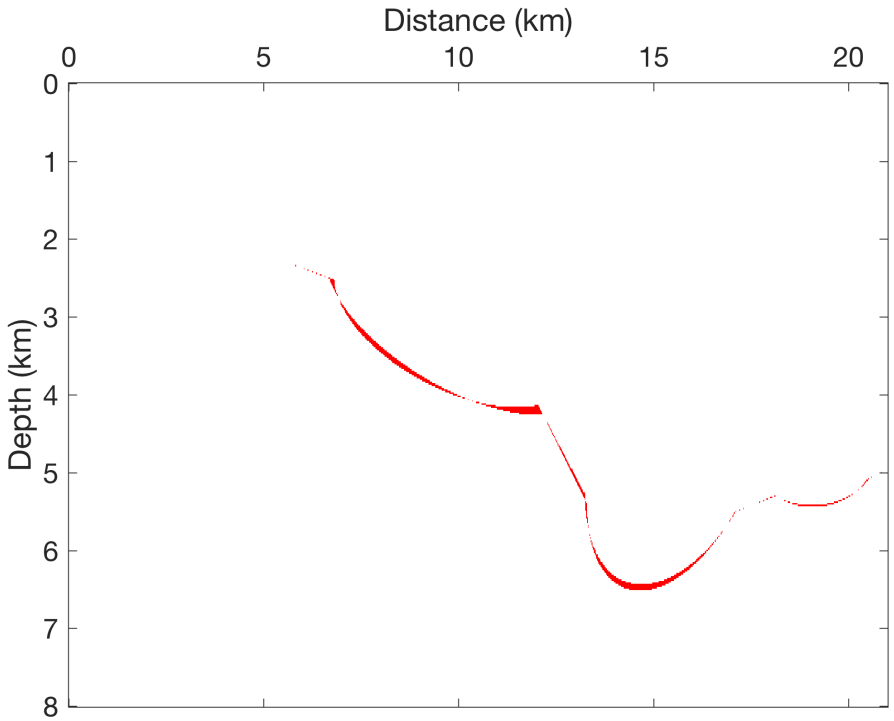


Figure 3.33: The red part shows the part of the salt not covered by all five realizations of VFSA run for the Sigsbee2A model. The uncertainty is pretty minimal compared to the salt body.

3.5.2 Uncertainty from inaccurate salt velocities

In my tests, I assume that the salt velocity is known before inversion. However, it can be difficult to obtain the true salt velocity in practical problems. When the salt velocity is not known, one can use salt velocity as one of the parameters in VFSA. I attempted that for the SEG-EAGE salt model with a search space on the salt velocity between 4200 and 4800m/s where the true salt velocity was 4481m/s. The salt velocity was recovered within a 4 percent error with an accurate estimate of the salt shape.

3.5.3 Uncertainty from Concave Hull

Unlike convex hull, the concave hull of a set of points is not unique. It is determined by a parameter α_{hull} that defines the tightness of the resulting shape. Different values of α_{hull} result in slightly different shapes of the salt body. While this may seem as being a cause of uncertainty, my tests indicate that within a small variation of α_{hull}, the ellipse parameters adjust themselves so that there is little to no change in the estimated models.

3.5.4 Uncertainty from the chosen number of ellipse

The number of ellipses (which was fixed a priori) is a crucial parameter. An insufficient number of ellipses would result in coarse shapes of salt bodies. Although using a large number of ellipses has the potential of obtaining accurate shape of the salt body, it may be computationally expensive. The problem can be addressed by letting the number of ellipses be a variable. Formulating

such a trans-dimensional approach in VFSA is fairly straightforward.

3.6 Conclusions

I demonstrated a new approach to estimate sediment velocities and salt shape using a combination of global and local optimization methods. I obtained good starting models by estimating sediment velocities using interfaces and velocities across them while a rough estimation of the salt shape is obtained by estimating an optimal combination of ellipses. The estimated model from global optimization and sparse parameterization does not suffer from cycle skipping in the seismic bandwidth and serves as a starting model for a gradient-based inversion. I use level-set methods to update the shape of the salt while the sediments are updated by conventional gradient methods. I also show that the VFSA models are consistent with different starting guesses and converge to similar solutions with multiple runs. The level-set method also works well in recovering the true salt shape from gradient information. The proposed method is semi-automatic and negates the need for several runs of migration with manual salt picking. The results of three different synthetic salt models of varying complexity demonstrate the usefulness of my approach.

Chapter 4

Global 3D Full Waveform Inversion with sparse parameterization

4.1 Summary

Estimating a starting velocity model for Full Waveform Inversion can be challenging. It requires several passes of migration velocity analysis to obtain a model accurate enough to prevent cycle skipping. I present an alternative approach to estimate a 3D starting model using a global optimization method called Very Fast Simulated Annealing. To constrain the optimization problem with a large number of unknowns, I parameterize the 3D model with surfaces and velocities surrounding them and solve for the optimal parameters. The final estimated model from VFSA serves as a starting model for FWI. I demonstrate the effectiveness by comparing FWI results along a few 2D lines from the 3D SEG-EAGE overthrust model. The proposed method is largely automated and reduces numerous man-hours required to build a starting model. I apply the proposed method to one toy model and one complex

Parts of the chapter were first published in *Datta, D., Sen, M. K., Morton, S., & Liu, F. (2017). Global 3D acoustic Full Waveform Inversion using sparse model parameterization. In SEG Technical Program Expanded Abstracts 2017 (pp. 1599-1603). Society of Exploration Geophysicists. The work was done by Debanjan Datta under a project funded by Hess Corporation.* The full version of the paper is under preparation.

synthetic model. In both cases, I am able to obtain acceptable starting models. Use of VFSA with a sparse parameterization method makes the 3D global inversion a practical tool.

4.2 Introduction

Building a starting model for Full Waveform Inversion (Tarantola, 1984; Virieux and Operto, 2009) requires enormous man-hours where an initial estimate of the velocity model is developed after multiple iterations of Migration Velocity Analysis (MVA) before arriving at a model that does not suffer from cycle skipping in the seismic bandwidth. This problem is further exacerbated in 3D where each migration pass is several orders of magnitude more expensive than that of 2D.

Several approaches have been proposed over the years to alleviate the problem of cycle skipping. The multiscale approach (Bunks et al., 1995; Sirgue and Pratt, 2004) steps over multiple frequencies to go from low to high frequencies. An alternative implementation in the Laplace Fourier domain was presented by Kim et al. (2013), which focused on different scales using damping values for the Laplace domain. Almomin et al. (2012) presented a composite objective function containing updates to both migration as well as tomographic components and thereby being less dependent on the starting model. Methods based on signal processing constitute adaptive filters proposed by Warner and Guasch (2014), Dynamic Image warping proposed by Ma and Hale (2013) or auxiliary Bump Functional by Bharadwaj et al. (2016). Another proposed

method is a hybrid optimization approach (Datta et al., 2016; Datta and Sen, 2016) where the starting model for FWI is estimated by using a combination of sparse parameterization in a global optimization method. Global methods are not strongly dependent on the choice of the starting model and because of sparse parameterization, they converge in finite iterations. Once a starting model is estimated, it is used in the conventional FWI.

In this chapter, I describe an extension of the velocity interface method to three dimensions. Following Datta and Sen (2016), I employ a sparse parameterization technique where I represent a velocity model using a set of interfaces and velocities across the interfaces. The idea behind the approach is to find an optimal set of interfaces and velocities in which the corresponding model has the minimum misfit using a global optimization technique. Once the model is estimated, I test it for correctness using the conventional FWI over a few 2D slices of the model.

4.3 Theory

Consider a 3D velocity model defined by $v(x, y, z)$. I define a set of interfaces in 3D as $z_i(x, y)$, where z_i is the depth of the interface at (x, y). Each interface is in turn parameterized by a set of isodepth contours defined as point sets. The collection of point-sets are then interpolated to obtain an interface defined in (x, y). Two velocities v_{up} and v_{dn} are defined at the top and bottom of each interface. When the set of interfaces and the velocities across them are decided, the complete velocity model is built by linearly interpolating

the velocities across them. This sparse parameterization allows us to represent a 3D model using a few finite parameters. To compute the seismic response of the models, I use the acoustic wave equation in 3D given by

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 P(\mathbf{x})}{\partial t^2} = \nabla^2 P + s\left(x, y, z, t\right), \tag{4.1}$$

where P is the pressure wavefield, ∇^2 is the Laplacian given by $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial z^2} + \frac{\partial^2}{\partial y^2}$, $v(\mathbf{x})$ is the velocity field and $s(x, y, z, t)$ is the source term. I sample the pressure wavefield at the receiver locations to obtain the desired seismograms. The seismograms are then compared with the recorded seismograms using a cross-correlation objective function.

I find an optimal set of velocities and interfaces by minimizing the misfit between the data from the true 3D model and data from random models. Very Fast Simulated Annealing (VFSA) (Ingber and Rosen, 1992) was used for this purpose. A starting model is updated iteratively using a control parameter called temperature and unlike greedy algorithms, is able to accept worse solutions. A detailed overview of VFSA can be found in Sen and Stoffa (2013).

Using the VFSA derived model along with several 2D slices, I carry out FWI. I use the same equation as in equation 6.1. To compute the gradient I back propagate the data residuals using the adjoint state method (Plessix, 2006) given by

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2 R}{\partial t^2} = \nabla^2 R + \Delta d\left(x, t\right), \tag{4.2}$$

where R is the adjoint wavefield , and $\Delta d\,(x,t)$ is the data-residual.

The gradient is computed by cross-correlating the forward wavefield with the adjoint one as

$$\frac{\partial E}{\partial v(\mathbf{x})} = \frac{-2}{v(\mathbf{x})^3} \sum_{shots} \frac{\partial^2 P}{\partial^2 t} R. \tag{4.3}$$

The gradient is now used to update the model using a L-BFGS optimizer (Zhu et al., 1997) to obtain the final model update.

4.4 Results

4.4.1 Validation test on toy model

First I demonstrate a proof of concept using a flat toy model with 4 layers and 3 interfaces shown in Figure 4.1. Because it is a model with flat interfaces, all shot gathers will give the same response. So the inversion was run with just one shot gather in the center of the model. I ran the VFSA algorithm for 100 iterations to obtain an optimal set of interfaces and velocities. The true values of the depth of interfaces and the velocities, their search space, and inverted values are shown in Table 4.1. One can observe that even with a wide search space, we are able to obtain a model that is very close to the true model.

4.4.2 SEG EAGE Overthrust Model

The second model I used was a modified EAGE-SEG 3D Overthrust model (Aminzadeh et al., 1997b) shown in Figure 4.2a. I selected half the

Toy model inversion			
	v_{up}	v_{dn}	Depth
Int 1	1500 [1500] (1500, 1500)	2004 [2000] (1750, 2250)	50 [50] (50, 50)
Int 2	2009 [2000] (1700, 2300)	2508 [2500] (2200, 2800)	98 [100] (75, 125)
Int 3	2520 [2500] (2200, 2800)	3021 [3000] (2800, 3500)	157 [150] (120, 180)

Table 4.1: Table showing VFSA results for a 4-layered toy model with 3 interfaces. The true values are shown in square brackets and the search space is given in parentheses

grid points from the strike direction making the model size (801x401x207) in the x, y and z directions respectively, each with a grid spacing of 15m. I generated synthetic data using a Ricker wavelet with a central frequency of 4Hz. The maximum offsets were 9 km and 4.5 km in the dip and strike directions respectively. To obtain an initial guess for the interfaces, I heavily smoothed the model and extracted a few surfaces with velocities starting from 2500 to 4500m/s. For a real dataset, this can also be done after semblance-based velocity analysis. I selected 5 interfaces from the smoothed model at velocity values between 2200m/s and 6000m/s. From each interface, set of a few points are identified. The search space of the few iso-depths was put 500m top and below the initial iso-depth. VFSA was run for 100 iterations; the final model is shown in Figure 4.2b. The error vs iteration plot is shown in Figure 4.3. The curve shows that error oscillates in the initial iterations while in the later iterations it searches for a better solution in the vicinity of the current solution. To demonstrate the quality of inversion, I show the shot gathers from the true model vs the VFSA model in Figure 4.4. The gathers show that they do not suffer from cycle skipping and, therefore, the derived model can potentially be a good starting model for standard FWI.

I took 2 slices from the 3D VFSA model from the (x, z) plane at y= 3km and 5.7km, and performed a 2D FWI on each of the slices. The inversion was done at frequency ranges of 3,4,5,6 Hz for 30 iterations per frequency. The slices from the VFSA model are shown in Figures 4.5b and 4.6b while the final models are shown in Figures 4.5c and 4.6c. The final models show

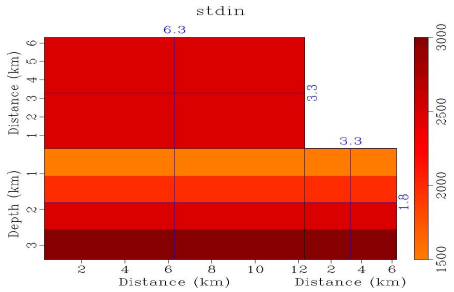
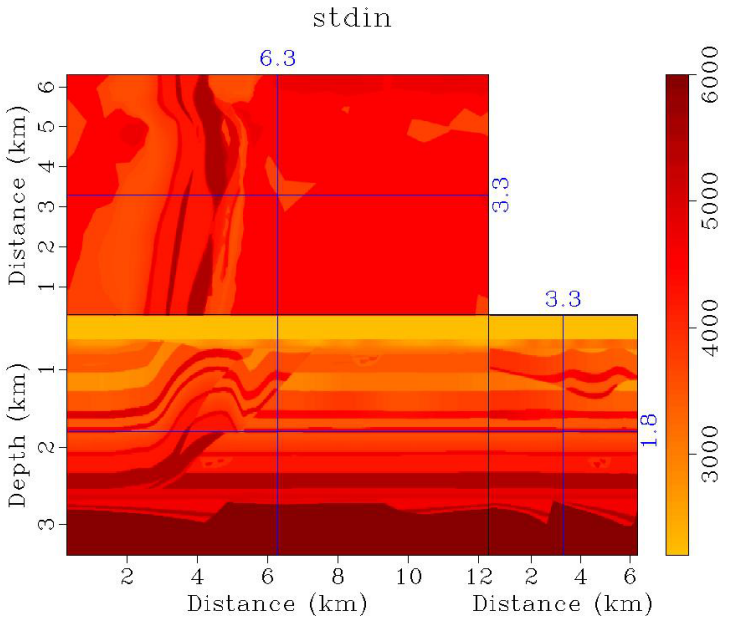


Figure 4.1: The 4 layer TOY model

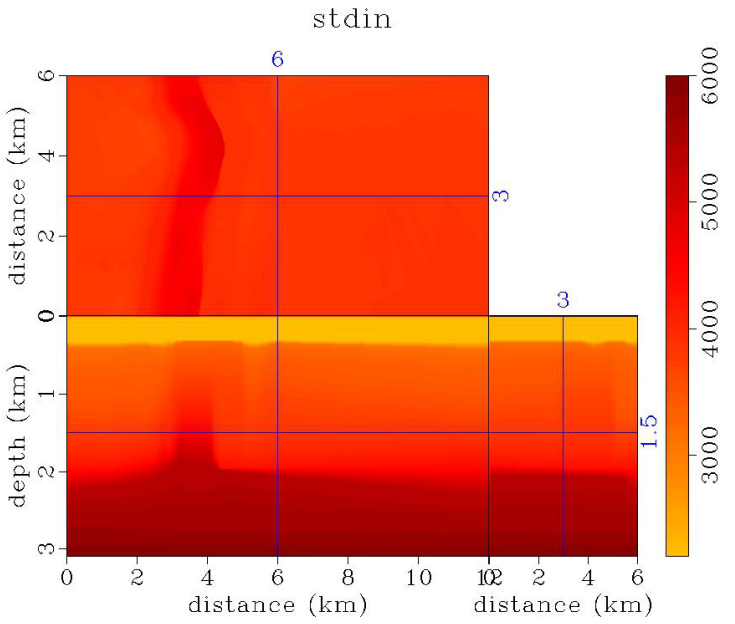
close resemblance to the true models shown in Figures 4.5a and 4.6a thereby demonstrating the effectiveness of the VFSA approach.

4.5 Conclusions

I present a novel FWI method using VFSA to estimate starting models in 3D. The method is largely automated and has the potential to automate velocity model building circumventing on a few iterations of Migration Velocity Analysis. The algorithm is able to reliably estimate the correct set of interfaces and velocities the corresponding model of which gives the least misfit with the real data. The estimated model does not suffer from cycle skipping and is able to recover the true model after conventional FWI. The cost of this approach is comparable to standard FWI as it avoids extraneous forward modeling operations to compute gradient and step lengths.



a



b

Figure 4.2: (a) True 2D SEG-EAGE Overthrust Model (b) Inverted model after 100 iterations of VFSA

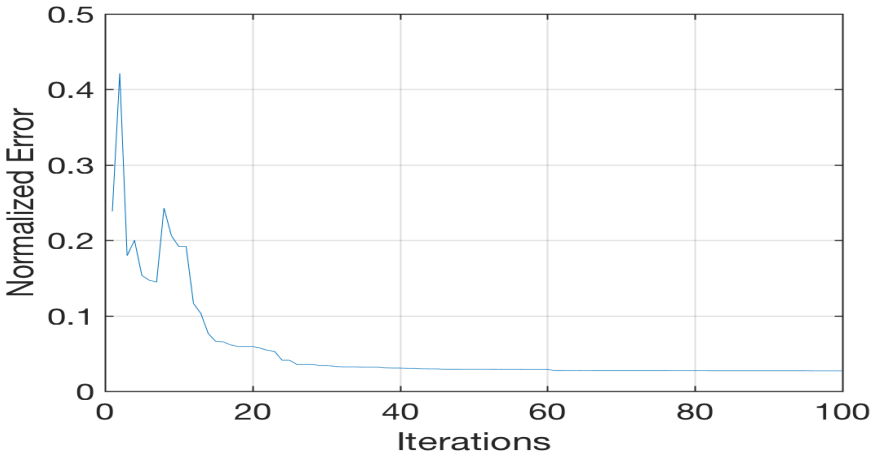
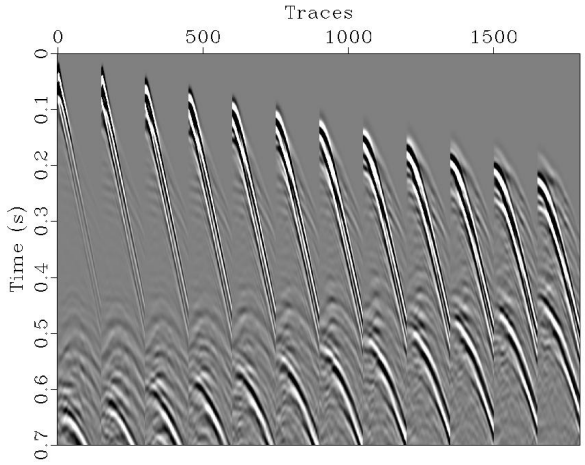
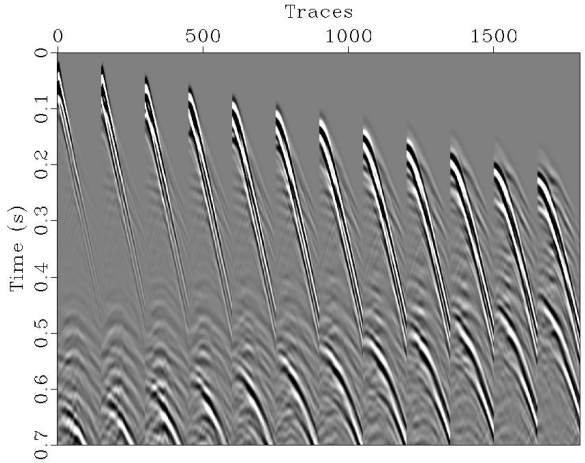


Figure 4.3: Error vs iterations for the SEG EAGE Overthrust Model

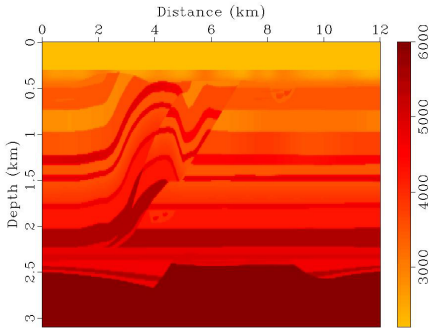


a

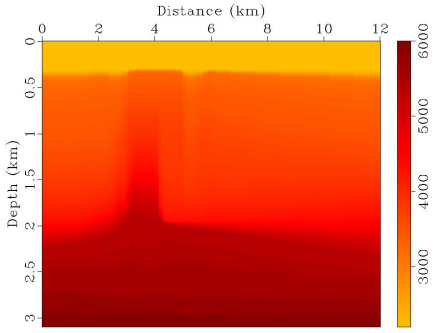


b

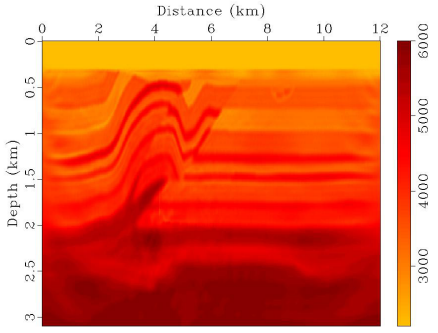
Figure 4.4: (a) Modeled Seismogram from the true SEG-EAGE overthrust model and (b) Modeled seismogram from the VFSA result



a

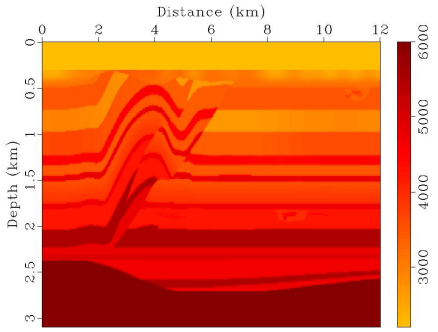


b

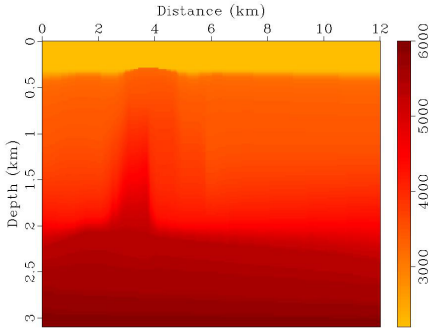


c

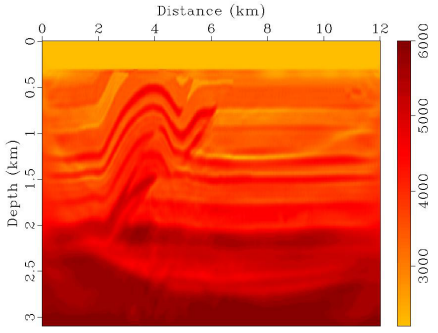
Figure 4.5: (a) The True 2D slice from the SEG-EAGE Overthrust Model at $y = 3.5km$ (b) Starting slice from VFSA and (c) Inverted 2D model from (b) at four frequencies with 30 iterations per frequency of gradient FWI



a



b



c

Figure 4.6: (a) The True 2D slice from the SEG-EAGE Overthrust Model at $y = 5.7km$ (b) Starting slice from VFSA and (c) Inverted 2D model from (b) at four frequencies with 30 iterations per frequency of gradient FWI.

Part II

Computational Issues in FWI

Chapter 5

Fast 2D Full Waveform Inversion in the frequency domain using Schur Complement Approach

5.1 Summary

Full Waveform Inversion (FWI) makes iterative updates to a starting model until a satisfactory match is obtained between the modeled and recorded data. The gradient in FWI is computed using the adjoint state method where two forward evaluations are used to compute the gradient of the misfit function. Additionally, computation of the step length requires a few more forward evaluations. The forward evaluation step is the most expensive part of the workflow, and running it multiple times makes FWI one the most computationally expensive workflows in the seismic industry. In most FWI problems sufficient prior information is available to constrain the velocity of

Parts of the chapter were first published in *Jaysaval, P., Datta, D., Sen, M., & Arnulf, A. (2017). A Schur complement based fast 2D finite-difference multimodel modeling of acoustic wavefield in the frequency domain. In SEG Technical Program Expanded Abstracts 2017 (pp. 4086-4090). Society of Exploration Geophysicists.* and in *Datta, D., Jaysaval, P., Sen, M., & Arnulf, A. (2017). Fast Full Waveform Inversion using a Schur-complement based frequency-domain finite-difference modeling. In SEG Technical Program Expanded Abstracts 2017 (pp. 1643-1647). Society of Exploration Geophysicists.* The full version of the paper has been submitted in Geophysics and is currently under review. The modeling and inversion codes were written by Debanjan Datta and the Schur subroutine was written by Debanjan Datta & Piyoosh Jaysaval.

certain parts of the model. Constrained parts may include the water column velocity, which is often unchanged during inversion. I propose a method where I partition a velocity model into two parts. The first part consists of the constrained zone that does not change and the second part is the domain that is iteratively updated during inversion. I pre-compute the contribution of the constrained part just once in the form of a Schur complement, and for every subsequent model evaluation, I update the Schur complement using the new anomalous velocities. Because the evolving part of the model is comparatively smaller than the whole model, considerable savings in wall clock times can be obtained using my partition approach. I describe the modeling approach using a marine model with varying water column depth and then demonstrate my results on FWI workflow over a reference model and the Sigsbee2A model.

5.2 Introduction

The crux of Full Waveform Inversion (FWI) (Tarantola, 2005; Virieux and Operto, 2009) consists of repeated forward evaluations to compute the gradient and step length during every iteration. When computing the gradient of the objective function using the adjoint state method (Plessix, 2006), two forward evaluations (one for forward propagation of the source and the other for backward propagation of the residual) are needed. Additionally, two or more forward evaluations are needed based on the method selected to approximate the step length (Wright and Nocedal, 1999). The forward modeling in FWI is usually implemented using Finite Difference methods (Crase et al.,

1990; Levander, 1988; Virieux, 1986), Finite Element methods (Choi et al., 2008) pseudo-spectral methods (Wu and Alkhalifah, 2014) or spectral-element methods (Capdeville et al., 2005; De Basabe and Sen, 2007; Magnoni et al., 2014) both in time domain (Crase et al., 1990; Tarantola, 1984) or in the frequency domain (Pratt and Worthington, 1990). Though both approaches have their merits and demerits, one common characteristic of forward evaluations is that they are computationally expensive. Performing a few forward evaluations for a large number of iterations makes FWI computationally prohibitive to use on a regular basis. The issue is exacerbated during multi-scale FWI (Bunks et al., 1995) where frequency stepping is performed across multiple runs of FWI.

Several techniques have been proposed to reduce the computational footprint of FWI. Shot encoding (Krebs et al., 2009; Morton and Ober, 1998; Schuster et al., 2011) allows combining multiple shot gathers to obtain a supergather and computing the misfit and gradient of multiple shot gathers using one forward evaluation. However, simultaneous sources are plagued with crosstalk between multiple shots when computing the gradient in FWI. Nevertheless, multiple approaches of crosstalk reduction (Godwin and Sava, 2011; Guitton and Díaz, 2012; Romero et al., 1999; Soubaras, 2006; Stoffa et al., 2006; Sun et al., 2002) have also been studied with favorable elimination of cross-talk. The injection grid method was proposed by Robertsson and Chapman (2012) in the time domain that injects wavefield values in the boundary of an anomalous zone and performing FD simulations in a smaller zone. This

method was demonstrated to work well in a time-lapse case study. Another approach to reduce computational expense of FWI is Downward Continuation (Arnulf et al., 2011) where sources and receivers are downward extrapolated on the seafloor reducing the computational cost significantly in a deepwater environment. The last category of methods includes skipping a few shots which do not significantly alter the FWI gradient. Two different approaches to this are random shot selection proposed by Díaz and Guitton (2011) and cyclic shot selection proposed by Ha and Shin (2012).

An extensive amount of computation and human effort goes into the development of the starting model for FWI. Traveltime tomography (Plessix and Perkins, 2010) is used to estimate starting models for FWI for a deepwater ocean bottom data. Migration Velocity Analysis (Sava and Biondi, 2004a,b) using different methods of seismic migration gives kinematically correct starting models in FWI and Prieux et al. (2013) used stereotomography to estimate starting models for FWI. These methods make use of model constraints such as geological constraints and near-surface velocity to estimate a starting model that does not suffer from cycle skipping in the seismic bandwidth and converges rapidly. Incorporating realistic prior information is important to obtain robust starting models, which in turn allows the user to constrain parts of the velocity model that are not updated during the inversion process as the prior information is enough to estimate it correctly. This leads to redundant computation of the response of the constrained zone multiple times during the FWI workflow across several iterations. In Deepwater models, the constrained

zone can be large enough (eg. water column velocity) to affect the runtimes significantly.

A standard (Frequency Domain Finite Difference) FDFD approach would demand discretizing the entire computational domain for every new velocity model even if the velocities change within a small zone. Here, I propose a new efficient modeling approach in the frequency domain, which is able to exploit the redundancy of constrained zones (Jaysaval et al., 2014). In this chapter, I refer to the changing velocity zone as the anomalous zone and the fixed velocity as background throughout the text. In the first forward simulation, the unknowns of the background are separated and their contribution is computed using a Schur complement. The computation of the Schur complement is done only once. To compute the effect of the evolving zone, the comparatively smaller Schur complement system is solved. For every new velocity model in subsequent iterations of FWI, the Schur complement is updated according to the new and updated anomalous zones by simple algebraic operations using the current Schur complement with the old and new anomalous sub-domain matrices. Therefore, each subsequent forward evaluation solves a comparatively smaller modified Schur complement system which leads to significant reduction in the computational cost of FWI.

I demonstrate the usefulness of the approach by showing a modeling study and establishing accuracy metrics for a simple model with varying water depth. The reduction in wall clock times is shown for 1) reference model and 2) Sigsbee2A model. I use the MUMPS direct solver (Amestoy et al., 2000) to

compute the Schur complement of the background subdomain and to solve the corresponding Schur complement system for the anomalous subdomain. The next section describes my approach in detail and I follow them with results and conclusions.

5.3 Theory

The constant density acoustic wave equation in the frequency domain is given as

$$\nabla^2 P(\mathbf{x}) + \frac{\omega^2}{v^2(\mathbf{x})} P(\mathbf{x}) = s(\mathbf{x}, \omega), \tag{5.1}$$

where, $P(\mathbf{x})$ is the pressure wavefeld, $\mathbf{x}$ is the co-ordinate vector, $v(\mathbf{x})$ is the P wave velocity, $s(\mathbf{x}, \omega)$ is the source, and ω is the angular frequency.

The synthetic data is recorded as

$$\mathbf{d_{syn}} = G(P(\mathbf{x})), \tag{5.2}$$

where $\mathbf{d_{syn}}$ is the data vector and G is the interpolation and extraction operator that extracts the wavefields at the specified receiver locations.

The cost function E is then computed by taking the difference between the observed data and synthetic data:

$$E = \frac{1}{2}(\mathbf{d_{syn}} - \mathbf{d_{obs}})^T(\mathbf{d_{syn}} - \mathbf{d_{obs}}), \tag{5.3}$$

where $\mathbf{d_{obs}}$ is the observed data and ^{T} is the complex conjugate.

The gradient of the cost function is computed by back propagating the difference of the observed and synthetic data $\Delta\mathbf{d}$ using the Adjoint State method (Plessix, 2006; Tarantola, 1984, 2005). The back-propagated wavefield is given by

$$\nabla^2 R(\mathbf{x}) + \frac{\omega^2}{v^2(\mathbf{x})} R(\mathbf{x}) = \Delta d(\mathbf{x}, \omega) \tag{5.4}$$

where, R is the adjoint wavefeld and $\Delta d(\mathbf{x}, \omega)$ is the data misfit. The gradient is computed by a zero-lag crosscorrelation of the forward and adjoint wavefields given by

$$\frac{\partial E}{\partial \mathbf{v}} = Re \sum_{shots} \frac{-2\omega^2}{v(\mathbf{x})^3} R(\mathbf{x}) P(\mathbf{x}) \tag{5.5}$$

where, Re denotes the real part.

Once the gradient has been computed, one can use it in a minimizing loop using Steepest Descent (Fleury and Perrone, 2012) , Conjugate gradient (Vigh and Starr, 2008), L-BFGS (Métivier et al., 2015) or Newton methods(Métivier et al., 2013).

I now focus on equation 5.1 and present my schur complement approach. Equation 5.1 can be recast into a system of solutions given by

$$\mathbf{Mp} = \mathbf{s}, \tag{5.6}$$

where, $\mathbf{M}$ is a discrete form of $(\nabla^2 + \frac{\omega^2}{v^2(\mathbf{x})})$ and $\mathbf{s}$ is the source. $\mathbf{M}$ is a square sparse matrix with a banded structure as shown in Figure 5.1 containing N rows and columns. N is the total number of grids nodes in the discretized

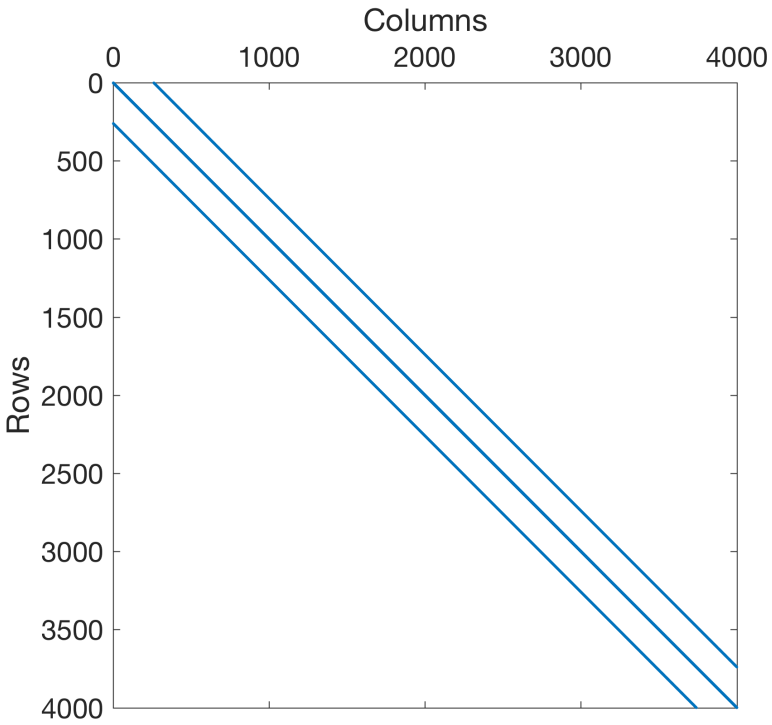


Figure 5.1: A typical Mass matrix showing banded structure. The matrix is square in shape with each dimension being N and the blue dots are the non zero elements while the white background represents zeros.

velocity model. Let a typical velocity model which one is trying to invert contain a fixed background (e.g a water column) and a changing anomalous (sub seafloor) zone as shown in Figure 5.2. I mark the water column as b and the sub-seafloor zone as a. Partitioning the velocity model in two zones also splits the banded matrix $\mathbf{M}$ into four blocks shown in Figure 5.3.

Writing equation 5.6 in partitioned block form I obtain

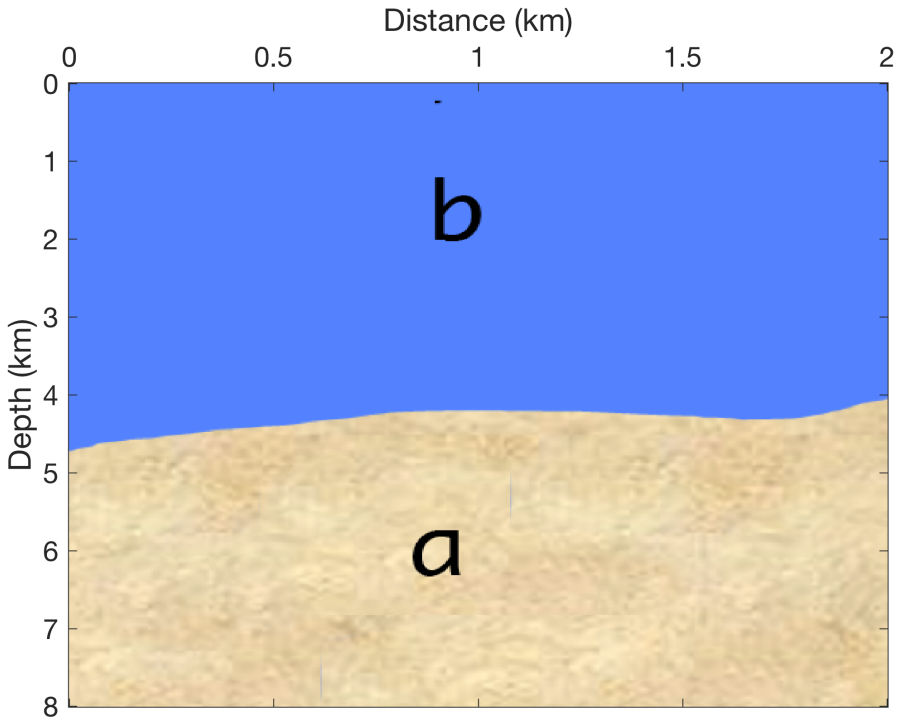


Figure 5.2: A prototype model being partitioned into background zone b which in this case is the water column. The background is not updated during inversion. The anomaly or sub-sea sediments are represented by a which is updated during inversion.

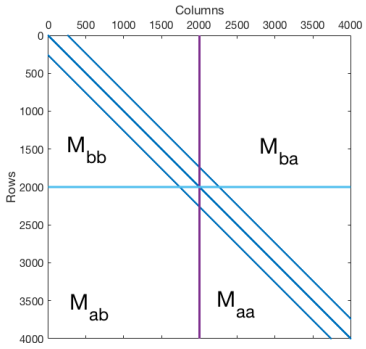


Figure 5.3: The partitioned matrix $\mathbf{M}$ being divided into 4 parts. The four parts represent the contribution of the background and anomalous zones to the propagating wavefield.

$$\begin{bmatrix} \mathbf{M}_{bb} & \mathbf{M}_{ba} \\ \mathbf{M}_{ab} & \mathbf{M}_{aa} \end{bmatrix} \begin{bmatrix} \mathbf{p}_b \\ \mathbf{p}_a \end{bmatrix} = \begin{bmatrix} \mathbf{s}_b \\ \mathbf{s}_a \end{bmatrix}, \tag{5.7}$$

where $\mathbf{M}_{bb}$ denotes the impulse kernel from the background,

$\mathbf{M}_{aa}$ denotes the impulse kernel from the anomaly,

$\mathbf{M}_{ab}$ denotes the impulse kernel from anomaly onto background,

and $\mathbf{M}_{ba}$ denotes the impulse kernel from the background onto anomaly.

I use a standard 5-point stencil for the second order in space finite difference discretization. The blocked system of equations shown in equation 5.7 is partially factorized only for $\mathbf{L}_{bb}$ eliminating the background unknowns to get

$$\begin{bmatrix} \mathbf{L}_{bb} & 0 \\ \mathbf{L}_{ab} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U}_{bb} & \mathbf{U}_{ba} \\ 0 & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{p}_b \\ \mathbf{p}_a \end{bmatrix} = \begin{bmatrix} \mathbf{s}_b \\ \mathbf{s}_a \end{bmatrix}, \tag{5.8}$$

where,

$\mathbf{M}_{bb} = \mathbf{L}_{bb}\mathbf{U}_{bb}$,

$\mathbf{M}_{ab} = \mathbf{L}_{ab}\mathbf{U}_{bb}$,

$\mathbf{M}_{ba} = \mathbf{L}_{bb}\mathbf{U}_{ba}$, and

$\mathbf{S} = \mathbf{L}_{aa}\mathbf{U}_{aa} = \mathbf{M}_{aa} - \mathbf{L}_{ab}\mathbf{U}_{ba} = \mathbf{M}_{aa} - \mathbf{M}_{ab}\mathbf{M}_{bb}^{-1}\mathbf{M}_{ba}$ and is the Schur Complement. The first term of the Schur Complement, $\mathbf{M}_{aa}$, is the impulse kernel of the anomalous block and the second term, $\mathbf{M}_{ab}\mathbf{M}_{bb}^{-1}\mathbf{M}_{ba}$ is the response of the background on the anomalous block.

Reformulating equation 5.7 using an intermediate vector $\mathbf{y}$ I get

$$\begin{bmatrix} \mathbf{L}_{bb} & 0 \\ \mathbf{L}_{ab} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{y}_b \\ \mathbf{y}_a \end{bmatrix} = \begin{bmatrix} \mathbf{s}_b \\ \mathbf{s}_a \end{bmatrix}, \tag{5.9}$$

where $\mathbf{y}_a$ and $\mathbf{y}_b$ are given by

$$\begin{bmatrix} \mathbf{U}_{bb} & \mathbf{U}_{ba} \\ 0 & \mathbf{S} \end{bmatrix} \begin{bmatrix} \mathbf{p}_b \\ \mathbf{p}_a \end{bmatrix} = \begin{bmatrix} \mathbf{y}_b \\ \mathbf{y}_a \end{bmatrix}. \tag{5.10}$$

I solve equation 5.10 as

$$\mathbf{S}\mathbf{p}_a = \mathbf{y}_\mathbf{a}, \tag{5.11}$$

$$\mathbf{U}_{bb}\mathbf{p}_b = \mathbf{y}_b - \mathbf{U}_{ba}\mathbf{p}_a. \tag{5.12}$$

I then solve equation 5.11 for the wavefield $\mathbf{p}_a$ in the anomalous zone a and use backward substitution in equation 5.12 to compute the wavefield $\mathbf{p}_b$ in the background.

One can now define a FWI workflow where velocities in the anomalous block are allowed to change while those in the background remain constant.

As a result, one obtains a new $\mathbf{M}$ at each iteration in which $\mathbf{M}_{bb}$ and hence $\mathbf{L}_{bb}$ and $\mathbf{U}_{bb}$ remain constant as the velocities in b are unchanged. Furthermore, because I use a standard finite difference grid, off-diagonal elements, $\mathbf{M}_{ab}$ and $\mathbf{M}_{ba}$ do not depend on the velocities and are invariant. This means $\mathbf{U}_{ba}$ and $\mathbf{L}_{ab}$ do not change. Following equation 5.10 one can note that $\mathbf{y}$ remains unchanged in the forward modeling stage since sources $\mathbf{s}_a$ and $\mathbf{s}_b$ do not change along the FWI iterative process. The adjoint sources, however, evolve during inversion, specifically $\mathbf{s}_b$. Therefore, I update the intermediate vector $\mathbf{y}_b$ at each iteration using forward substitution. Therefore, to find the response $\mathbf{p}^{'}$ from a new velocity model, one needs an updated Schur Complement $\mathbf{S}^{'}$ to obtain $\mathbf{p}_a^{'}$ using equation 5.12 and then use backward substitution to obtain $\mathbf{p}_b^{'}$. The updated Schur Complement $\mathbf{S}^{'}$ is given by

$$\mathbf{S}^{'} = \mathbf{S} - \mathbf{M}_{aa} + \mathbf{M}_{aa}^{'}, \tag{5.13}$$

where $\mathbf{M}_{aa}^{'}$ is the updated impulse response of the anomalous block.

Because the Schur Complement system in equation 5.11 is generally small, the Schur complement method is able to compute the wavefields faster than the standard FDFD method which has to solve the full system in equation 5.6 repeatedly over all iterations.

5.4 Results

I demonstrate the effectiveness of my approach using a simple model with varying water depth. I then use both FDFD and Schur Complement approach to compare FWI runtimes for a simple reference model and the Sigsbee2A model. All my simulations are carried out using 4 cores of an Intel Xeon CPU E5-1650 processors running at 3.5 GHz and 128 GB of system memory.

Forward modeling and accuracy test using a model with varying water depth

Here I use a model with varying water depths characterized by a 5 km-thick sub-sea sediment layer in which a 0.5 km-thick by 5 km-long low-velocity anomaly block has been inserted near its center (Figure 5.4). I obtain 4 different models by varying the thickness of the water column at 1, 2, 4 and 6 km. Each model was discretized at 10m intervals in both x and z directions. I modeled seismic wavefields for all 4 models using (1) standard FDFD and (2) the Schur complement approach. The forward modeling runtimes obtained using the standard FDFD method vs the Schur complement methods are shown in Table 5.1 and Table 5.2, respectively. Starting with the model with 6 km water depth, the standard FDFD takes 23.9s. The Schur complement is computed in 13.8s, and the Schur complement system is solved in 10.4s. This gives a 57% reduction in multi-modeling applications. Similarly, for the other models with water column depths of 4 km, 2 km, and 1 km, I obtained speed-ups

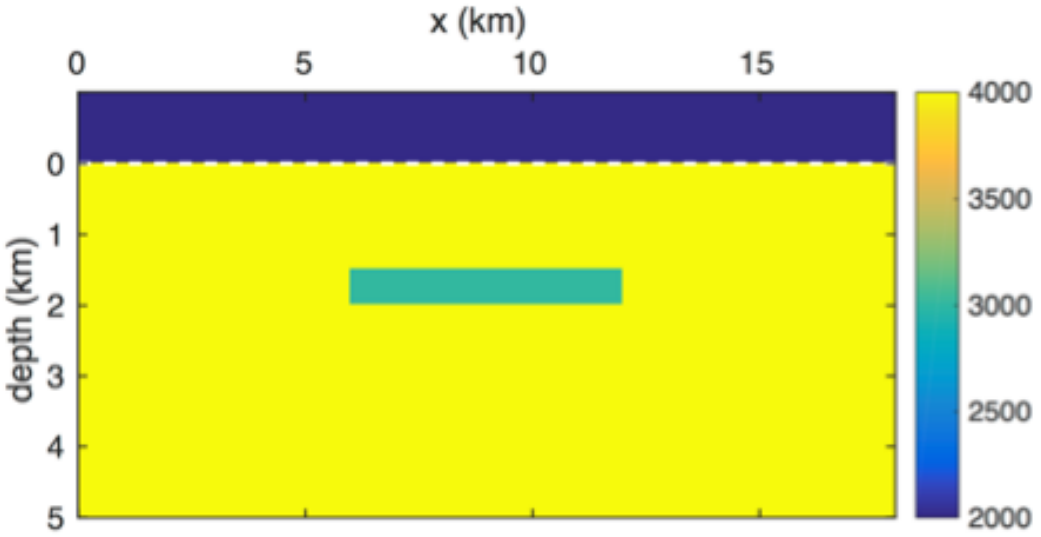


Figure 5.4: The two-layered model with a low velocity block in the middle. The length of the sea floor is variable ranging from 1 to 6 km.The model is discretized at 10m uniformly in vertical and horizontal directions. The horizontal span of the model is 20km while the vertical span ranges from 6km to 11km.

of 48%, 32% and 24%, respectively. One can conclude that the speedup is dependent on the ratio of the size of the background and anomalous zones; as the ratio increases, the speed-up also increases. Note that the Schur complement is not an approximation. It is just an algebraic rearrangement of the FDFD solver. The wavefields obtained from the Schur complement method is exactly equal to the standard FDFD method. I display the real part of the wavefields obtained by the standard FDFD method and the Schur complement method in Figure 5.5 and 5.6 respectively. The wavefields match exactly and the difference between them is zero.

5.4.1 FWI for a simple synthetic model

The synthetic model for FWI used in this example is shown in Figure 5.7a. The model consists of a two-layered half-space with a high-velocity block in the middle. The dimensions of the model are 800 points in the vertical z direction and 2010 points in the horizontal x direction with a grid spacing of 10m. The water column is 4km deep and is not updated during inversion. Below the water column, the velocities are updated during inversion. I describe the Schur interface at the sea floor, which is shown with a dotted white line in Figure 5.7a. The Schur interface partitions the model into the background zone that contains the water column and the anomalous zone that contains the sub-seafloor sediments. Across the 20km horizontal extent of the model, I generated data for 101 shots at an interval of 200m. The receiver pattern was fixed for all shots with 400 receivers placed at 50m intervals. Both the shots and receivers were placed at 20m depth.

I selected the two-layered half space without the high-velocity block as the starting model as shown in Figure 5.7b. The synthetic data were generated in 3 frequency groups between 3 and 7Hz, and the same were inverted using a multi-scale approach using 30 iterations per frequency group. The synthetic data was generated using a standard FDFD method. I used a L-BFGS algorithm to minimize the data misfit and timed the inversion run for standard FDFD as well as the Schur complement solver. Both the standard FDFD and Schur complement required 220 forward evaluations to complete the inversion. The inverted models obtained by the standard FDFD method and the Schur

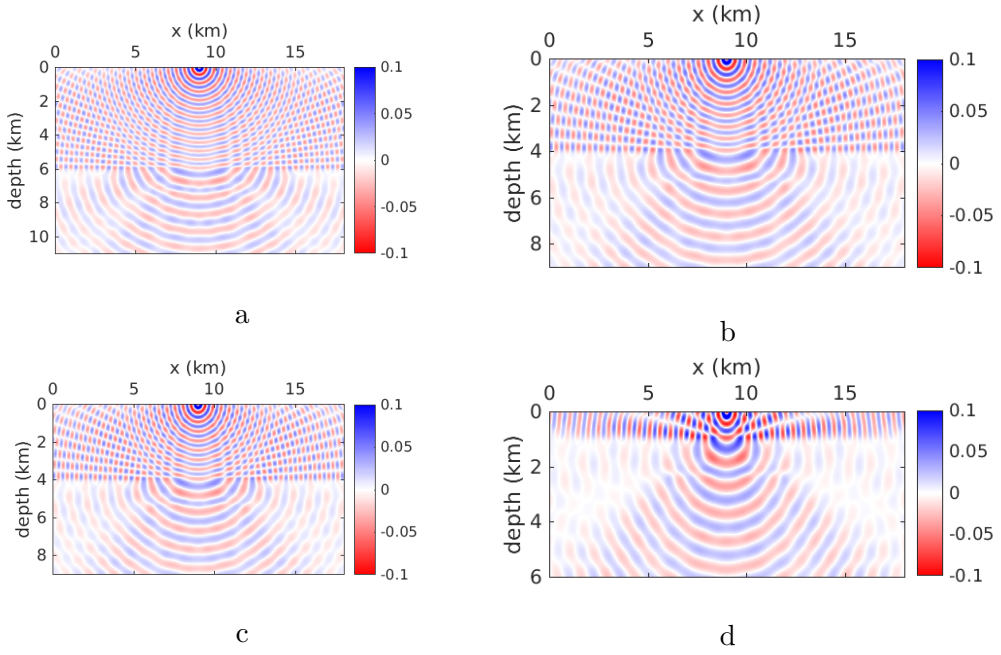


Figure 5.5: Seismic Response using standard FDFD of an identical 5 km-thick crustal model overlaid by varying water layer thicknesses W_d (a)W_d=6km (b)W_d=4km (c) W_d=2km and (d) W_d=1km.

Complement method are shown in Figure 5.7c and 5.7d, respectively. They are exactly identical velocity estimates just differing in their runtimes. The inverted model successfully reconstructs the high-velocity block missing from the starting model. The runtimes for both approaches are shown in Figure 5.8. Using the Schur approach, we were able to perform the inversion in about 40% of the time it took the standard FDFD to finish the inversion.

5.4.2 Sigsbee2A

The second model I used for the tests was the Sigsbee2A model shown in Figure 5.9a. The model contains 801 points in the vertical z direction and

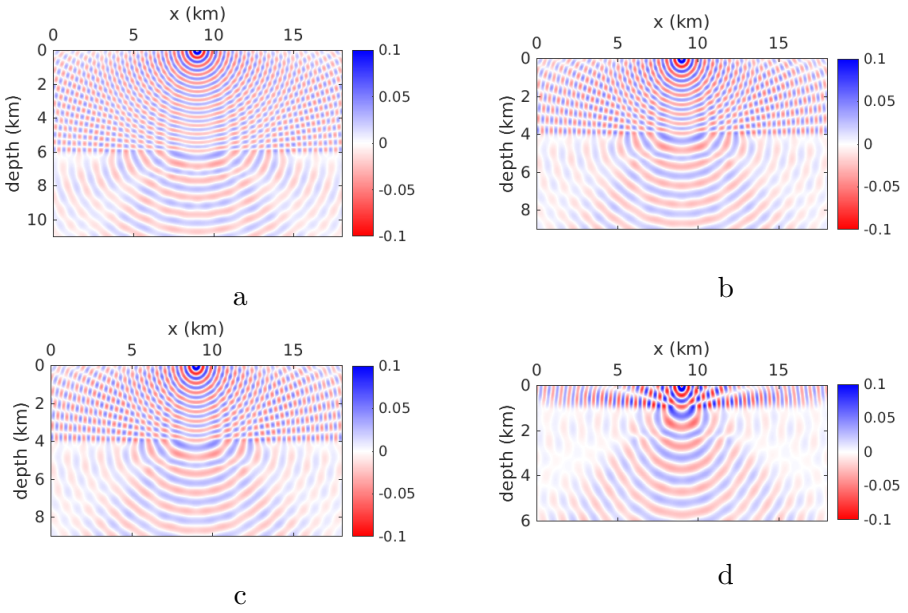


Figure 5.6: Seismic Response using Schur complement approach of an identical 5 km-thick crustal model overlaid by various water layer thicknesses W_d (a)W_d=6km (b)W_d=4km (c) W_d=2km and (d) W_d=1km.

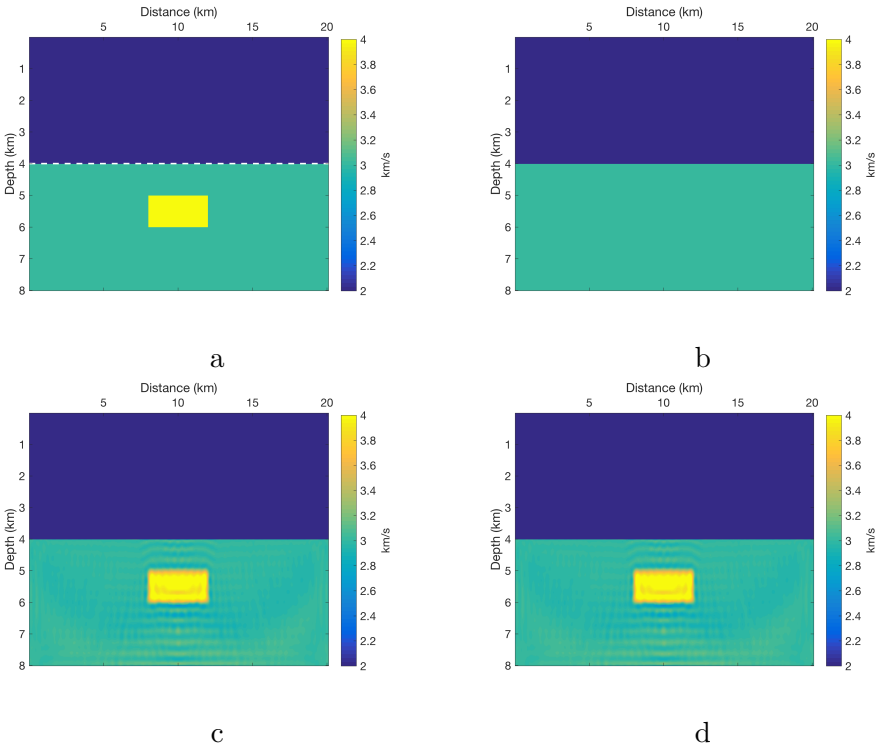


Figure 5.7: (a) True synthetic model for FWI, (b) Starting model omits the anomalous block from the true model, (c)Inverted model after 3 groups of multi-scale FWI using standard FDFD approach & (d) Inverted model after 3 groups of multi-scale FWI using the schur complement approach.

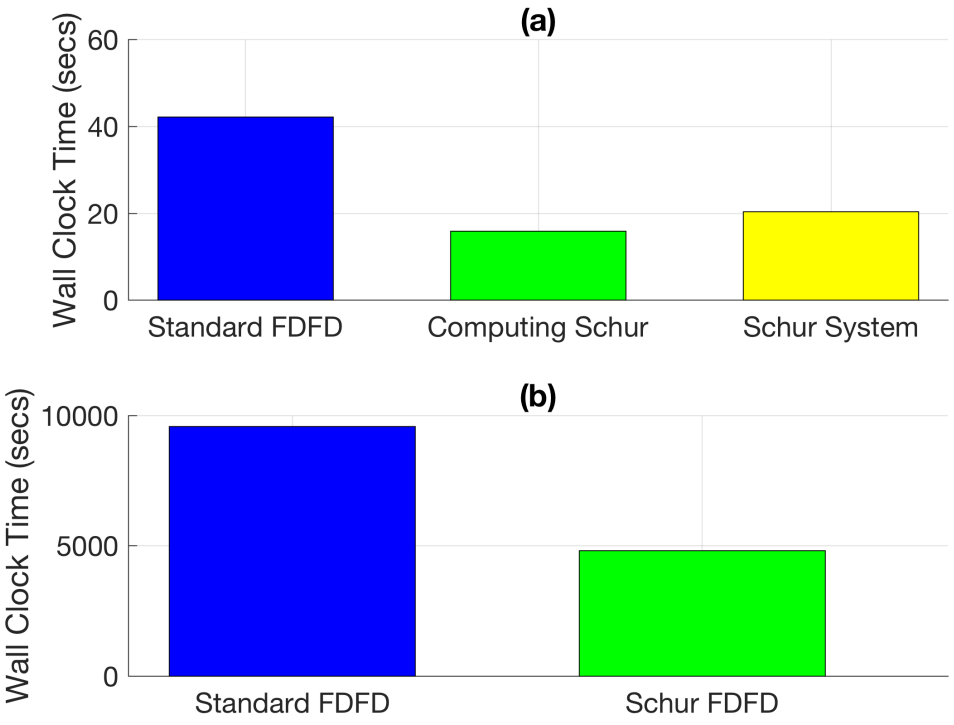


Figure 5.8: Comparison of run-times betwen standard FDFD and Schur Complement approach

2133 points in the horizontal x direction with a grid spacing of 10m. 267 shots were placed every 80m across the horizontal extent of the model. The receiver arrangement was fixed for all shots and 1065 receivers were placed 20m apart across the horizontal span of the model. Shots and receivers were placed at a depth of 20m. I generated synthetic data using 3 frequency groups between 3-8 Hz and selected the migration velocity for Sigsbee2A as the starting model in the FWI runs shown in Figure 5.9b. The migration model has the true salt shape and velocity and smoothed sediments. The sea-floor was matched exactly with the true model and the velocity of the water column was also the same as that in the true model. The Schur interface was defined at a depth of 1.6km as shown by the white dotted line in Figure 5.9a. The line was selected so that all grid points above this interface were part of the water column. The L-BFGS procedure was used to minimize data from each frequency group for all 30 iterations. A total of 251 forward evaluations were required to obtain the final inverted model. I show the inverted model from the standard FDFD approach as well as the Schur complement approach in Figure 5.9c and 5.9d, respectively. Similar to the other model, the inverted model is exactly identical only differing in runtimes. The inverted model successfully the stratigraphy of the sedimentary basin surrounding the salt body. The individual runtimes of the standard FDFD, solving Schur complement and obtaining the Schur complement system are shown in Figure 5.10. The Schur complement method was 20% faster than the standard FDFD method for the Sigsbee2A model.

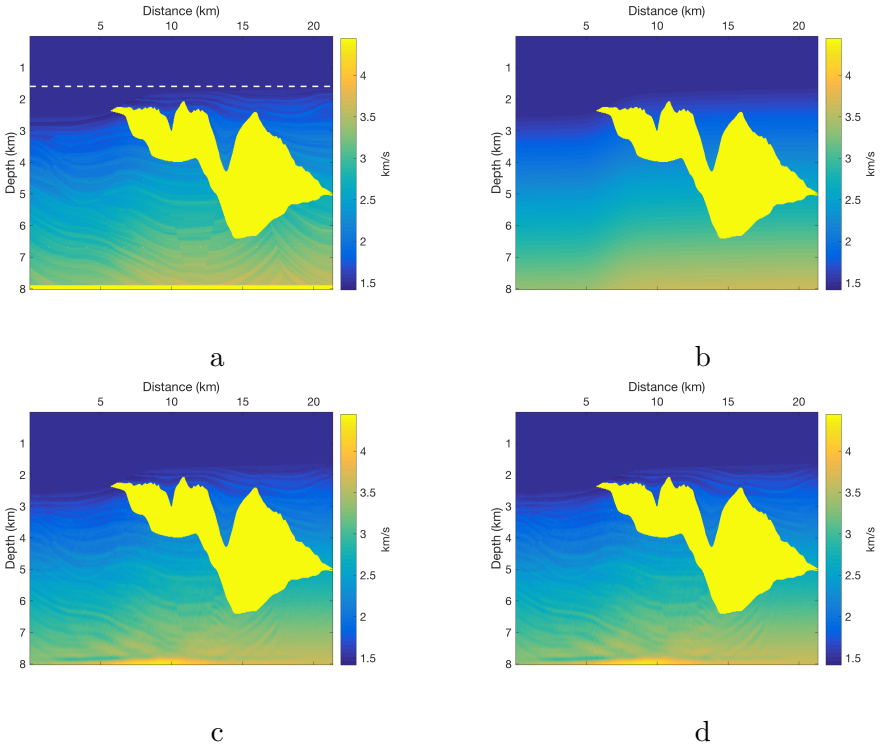


Figure 5.9: (a) True Sigsbee2A model for FWI (b) Starting model is the Sigsbee2A migration velocity (c) Inverted model after 3 groups of multi-scale FWI using standard FDFD approach & (d) Inverted model after 3 groups of multi-scale FWI using the schur complement approach.

Standard FDFD Modeling				
W_d (km)	N	NZ	Memory (GB)	Time(s)
6.0	2,157,600	10,781,960	3.9	23.9
4.0	1,785,600	8,922,360	3.1	19.8
2.0	1,413,600	7,062,760	2.5	15.3
1.0	1,227,600	6,132,960	2.1	13.6

Table 5.1: Runtimes and memory requirements per frequency for forward modeling using the standard FDFD approach.

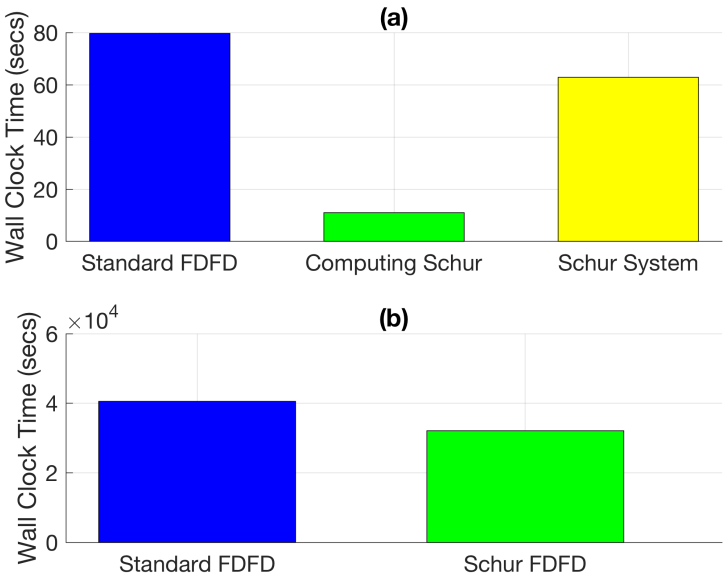


Figure 5.10: Comparison of run-times between standard FDFD and Schur Complement approach.

Schur FDFD Modeling								
Compute $\mathbf{S}$ by partial factorization of $\mathbf{M}$					Compute $\mathbf{x}_a$ by solving $\mathbf{S}\mathbf{x}_a = \mathbf{y} - a$			
W_d (km)	N	NZ	Memory (GB)	Time(s)	N	NZ	Memory (GB)	Time(s)
6.0	1,177,380	5,881,914	6.8	13.8				
4.0	805,380	4,022,314	4.9	9.7	985,800	8,378,242	1.8	10.4
2.0	433,380	2,162,714	3.2	5.5				
1.0	247,380	1,232,914	2.4	3.4				

Table 5.2: Runtimes and memory requirements per frequency for forward modeling using the Schur Complement approach

5.5 Discussions

I discuss a few points on the extendability of the Schur complement method to other formulations and dimensions.

5.5.1 Extension to elastic and anisotropic cases

In my formulation, the off diagonal elements are invariant because of using a standard finite grid stencil. However, this can also be achieved by shifting the Schur interface upward by a number of grid points equal to the radius of the finite difference stencil. Shifting the Schur interface upward by a few grid points makes the off-diagonal elements invariant making the Schur complement method extendable to all kind of wave equations.

5.5.2 3D extension

Moving to three dimensions instead of two does not change the fundamental nature of the problem. It only makes the matrix larger and changes the pattern with more off-diagonal strips.

5.5.3 Undulating seafloor

The case of an undulating seafloor requires a conformal mapping (Shragge, 2014) to flatten the seafloor with varying values of δz and δx. However, in this case, only the finite difference coefficients change. So, the Schur complement method can be used in the case of the undulating seafloor. Using an undulating Schur interface will yield better runtimes than that obtained by the flat

Schur interface used in the Sigsbee2A example.

5.6 Conclusions

I demonstrate a novel approach that is able to partition a velocity model into the background and anomalous zones. I pre-compute the contribution of the anomalous zone through a Schur Complement. All subsequent modeling operations are done by algebraically manipulating the Schur complement and the Schur complement system from the new anomalous block. Because the Schur complement system can be comparatively smaller, computing the Schur system alone can take significantly less time than that required by the standard FDFD approach. The savings over a single data modeling function add up over multiple iterations in FWI to give considerable savings in time-to-solution. The savings in compute time depends on the proportion of the anomalous and background blocks as shown in the examples. Additionally, because the approach is a modification to the solver itself, it can be combined with various multiple shot encoding methods mentioned in this chapter, which will reduce the runtimes even further.

Chapter 6

Efficient Gradient and Image Computation using a two level checkpointing method for adjoint problems using NVLINK connected CPU-GPUs

6.1 Summary

We noted, in Chapter 2, that the computation of gradient involves cross-correlation of the forward propagated wavefields with the back propagated data residual. The process of Reverse Time Migration (RTM) works in a similar fashion. This, however, requires the two wavefields from the same time to be available simultaneously. This creates a data reversal problem because the forward wavefields are generated in the opposite order of their access. Therefore, because of memory limitations, checkpointing methods are used for data reversal. Implementing checkpointing, however, further increases computations and data movement. In this chapter, I first present a useful model for the computational cost and memory requirements of two-level checkpointing with parameterizable values for system configurations (e.g. time to compute

A manuscript based on this work has been submitted to the Computing in Science and Engineering journal, Computer Society, IEEE. Debanjan Datta developed the framework with help from David Appelhans, Constantinos Evangelinos & Kirk Jordan.

a timestep, bandwidths to memory pools, memory pool sizes) to solve adjoint problems on GPUs. This model gives optimal choice for fine and coarse checkpoint frequencies, within the memory constraints of the system, and can aid in the comparison of various system configurations and choice of checkpoint frequencies. The proposed model is also parameterizable to future systems by just plugging in the updated bandwidth and compute performance parameters. Secondly, I evaluate the benefits of a fast NVLINK connection between GPU and CPU and the benefits of using an on-node Non-Volatile Memory Express (NVMe) memory pool vs traditional hard disks. I present numerical results for a GPU accelerated two-level checkpointing implementation for the Full Waveform Inversion (FWI) adjoint problem on different cluster configurations, including with and without NVLINK and NVMe. The proposed model matches observed runtimes and also shows good scaling characteristics.

6.2 Introduction

6.2.1 Overview

Adjoint computations are essential for problems like uncertainty quantification, data assimilation and derivative based optimization such as FWI (Virieux and Operto, 2009). The majority of these problems include time-dependent Partial Differential Equations (PDEs) to advance a state from zero time to final time, after which the adjoint states are computed in the reverse order from final time to zero time. Computations requiring the forward and adjoint states from the same point in time require a large (and often unavailable)

amount of memory to store the states after the forward run. This problem is addressed by using checkpointing strategies, which save only a subset of the forward states, and recomputes missing states as needed from the stored checkpoints, trading memory consumption with re-computation and minimal data movement. In the case of time stepping on GPUs, which are very fast but have relatively little memory compared to system RAM or on-node storage, the time to compute a time step is often significantly faster than the time required to move a checkpoint on or off the limited GPU memory. Without careful planning of checkpoint frequency, the GPU will wait for data movement to finish before being able to resume timestepping. This reduces GPU utilization and delays the time to solution. Similarly, when solving large 3D problems, the CPU memory often should be augmented with a larger but slower memory pool, resulting in another level of checkpointing that must be balanced against the time to recompute. A well optimized data movement strategy, such as the one presented in this paper, is able to take the compute and memory pool speeds into account to pipeline checkpointing and compute operations seamlessly, allowing for a shorter time to solution.

Large scale adjoint problems have been traditionally looked at as compute-intensive problems; however with advances in the speed of accelerators, it may be beneficial to rethink them as data intensive problems that need to be addressed with faster data movement. This is because an adjoint problem poses the data reversal issue, where data is generated in reverse order of access. Two current state of the art hardware technologies that facilitate faster

data movement between storage and compute devices are NVLink and Non Volatile Memory Express (NVMe). NVLink is the high speed interconnect between Nvidia GPUs, that also can link IBM POWER CPUs and their associated system RAM with Nvidia GPUs. Compared to PCIe, NVLink has a higher peak bandwidth of 40GB/s compared to 16GB/s for PCIe (Buono et al., 2017), meaning Nvidia GPUs connected to POWER processors now have a much faster connection to the large system RAM. NVMe is a very high speed storage technology that is significantly faster than spinning drives or SSDs with write speeds upto 1.3 GB/s for one thread.

In this paper, I investigate a large scale adjoint problem, namely Full Waveform Inversion (Virieux and Operto, 2009), and demonstrate how the time-to-solution can be greatly reduced by employing an optimal checkpointing strategy that exploits the faster speed of NVLink and NVMe. The faster interconnects facilitate faster data movement, and the implementation of overlapping compute and data movement combined with a smart choice of checkpointing frequency allows simultaneous utilization of compute resources, data pool bandwidths, and available storage space. All are critical to scaling to very large problems across multiple compute nodes with GPUs as accelerators.

The main contributions of this chapter are as follows

- Design an asynchronous two-level checkpointing method to fully hide data transfer when solving general adjoint problems.
- Propose a compute model parameterized by bandwidth and compute

times which is fully extendable to newer generation hardware.

- Propose a sub-optimal compute model when partial data transfer is hidden due to memory constraints.

- Highlight the benefits of a high speed interconnect between the CPU and GPU as well as well fast on node storage.

My proposed two-level checkpointing method stores and restores checkpoints at two levels, fine and coarse in different storage devices. I show that for optimal values of checkpointing parameters, one is able to hide all data movement behind compute obtaining a fully pipelined FWI engine.

6.2.2 Full Waveform Inversion

FWI is a critical workflow in seismic processing that uses the two-way wave equation to iteratively update a starting model to minimize mismatch with recorded data. Other algorithms, such as like Least Squares Reverse Time Migration Dai et al. (2012) and Reverse Time Migration Baysal et al. (1983) use similar kernel and the results from this paper apply directly to them with very few modifications. Often, it takes a few weeks to run a FWI workflow. In order to reduce the time-to-solution of a seismic workflow, it is imperative that the RTM/FWI workflows run faster because they take up a significant compute time in the seismic workflow.

The computational engine of FWI is a PDE solver that estimates the gradient of the cost function as described in chapter 1. I discuss the isotropic,

constant density acoustic wave equation, but the results apply to elastic as well as anisotropic formulations. The PDE solved in this case is the wave equation, which is given by

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2}{\partial t^2}P(t,\mathbf{x}) = \nabla^2 P(t,\mathbf{x}), \tag{6.1}$$

where $\mathbf{x}$ refers to the location in space, P refers to the pressure wavefield and t is the time. I solve equation 6.1 using a second-order finite difference discretization in time and 10th order discretization in space. The second order in time solver needs the previous and current pressure states to compute the next pressure as it marches forward in time. The data $\mathbf{d}(t,x)$ are recorded by extracting the values of the wavefields at sensor locations which are commonly placed on the surface.

For the adjoint solution, the data misfit is back propagated using the adjoint wave equation. The wave equation in this case is self adjoint and the back propagation is given by

$$\frac{1}{v^2(\mathbf{x})}\frac{\partial^2}{\partial \tilde{t}^2}R(\tilde{t},\mathbf{x}) = \nabla^2 R(\tilde{t},\mathbf{x}) + \Delta d(\tilde{t},\mathbf{x}), \tag{6.2}$$

where $R(\tilde{t},\mathbf{x})$ is the adjoint wavefield and Δd is the data misfit. The term $\tilde{t}$ indicates that the equation is solved backwards in time from $t = nt$ to $t = 0$. The reflectivity image is constructed by summing the zero-lag cross-corellation of the source and adjoint wavefields over multiple sources and time-steps. This gradient of the misfit function, g, is given by

$$g = \frac{-2}{v^3(\mathbf{x})}\sum_{i=1}^{i=nshot}\sum_{t=0}^{t=n_t}\ddot{P}(t,\mathbf{x})R(t,\mathbf{x}), \tag{6.3}$$

where $nshot$ is the number of source simulations, the $\ddot{}$ operator refers to second derivative in time and n_t is the number of time steps used to discretize the shot simulation. Using equations 6.1 & 6.4 we can rewrite g as

$$g = \frac{-2}{v^3(\mathbf{x})} \sum_{i=1}^{i=nshot} \sum_{t=0}^{t=n_t} \nabla^2 P(t, \mathbf{x}) R(t, \mathbf{x}). \tag{6.4}$$

Because the forward and adjoint wavefields are timestepped in different directions and the correlation calculation requires both forward and adjoint wavefields, it becomes imperative to have a way to access the forward wavefields in reverse order. This is represented in Figure 6.1. In a typical simulation, there would be thousands of red cubes representing the forward wavefield and saving only a subset of these states (checkpointing) is essential to solving a large RTM/FWI problem with practical disk and memory sizes.

6.2.3 Related Work

A few alternate approaches to address the problem of time reversal have been proposed. The boundary-saving technique proposed by Gauthier et al. (1986) exploits the self-adjoint property of most wave equation formulations, saving the boundaries at each time-step and recreating the forward wavefields back in time by re-injecting the boundary values at every time step. Clapp (2009) applied random boundary conditions that allowed for simple time-reversal of forward wavefields. However, in the case of non-self adjoint problems, such as the viscous wave-equation, the random boundary condition as well as the boundary saving technique do not work well. Checkpointing

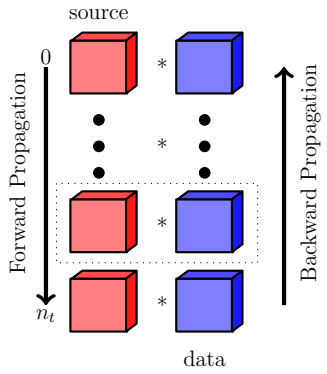


Figure 6.1: Forward propagating states (red cubes) and the backward propagating adjoint steps (blue cubes) in a FWI problem. For each point in time, the forward and backward states need to be cross-correlated as shown by the * sign. Since the forward states are generated in reverse order of their use, one needs to save a subset of the forward states from which one can regenerate all forward states when needed for cross-correlation.

does not depend on the problem being self adjoint, so it is a robust way to address general adjoint problems and with asynchronous data movement it can be quite viable.

While FWI/RTM kernels have been classically treated as a compute intensive problem (Ghosh et al., 2014; Narang et al., 2013; Sun and Suh, 2011), I address it as a data flow problem. A one-level checkpointing has been the preferred method to use in a FWI problem. In a one-level checkpointing scheme (Griewank and Walther, 2000; Sun and Suh, 2011) , One saves checkpoints at every few timesteps and reconstruct intermediate steps from the stored checkpoints. A pictorial representation for one-level checkpointing sheme is show in Figure 6.2. Because a checkpoint needs to be saved every few time steps, memory size becomes an issue when solving an adjoint prob-

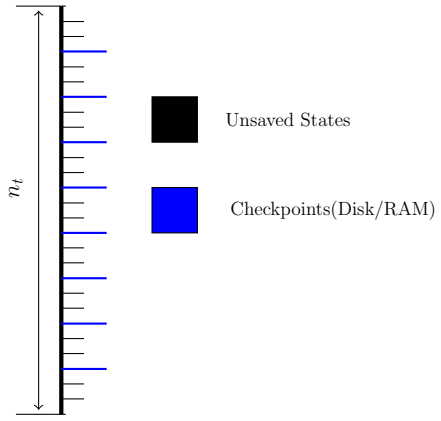


Figure 6.2: The vertical line represents the time scale from 0 to n_t in a one-level checkpointing scheme. Checkpoints are shown in blue horizontal lines while black horizontal lines show the intermediate steps which are not saved. During the forward pass all checkpoints are stored in a storage device.

lem with several thousand time steps. One of the two solutions to this is to move the checkpoints into a slower storage area. However slower data movement hinders computation and the processor is forced to wait to load/store the checkpoints. The other approach is to checkpoint less frequently which uses less memory but re-computations are increased by a large amount because of far-off checkpoints.

As a result, to scale to large problem sizes, one is forced to look at more complex checkpointing methods to work within memory constraints as well as to minimize re-computations. Multi-level checkpointing techniques (Bautista-Gomez et al., 2011; Moody et al., 2010) were introduced to deal with bigger data volumes. A synchronous two-level checkpointing has been proposed by Aupy et al. (2016), while an asynchronous version has been proposed by Schanen et al. (2016). However, both of there methods focused on CPU compu-

tations and did not include the additional level of GPUs or on-node memory in their checkpointing scheme. Also, Stumm and Walther (2010) proposed a multistage checkpointing approach to minimize access overhead to stored checkpoints. I exploit the hierarchical nature of current compute and storage devices and propose a two-level checkpointing method coupled tht exploits NVLink and NVMe to perform fully asynchronous data movement and compute.

6.3 Theory

I discuss a brief theory on two-level checkpointing approach and develop various performance models. The theory is not specific to FWI and applies to most adjoint problems.

6.3.1 Two Level Checkpointing

Assume an adjoint problem with n_t with order of the PDE being implemented be n_o. The two level checkpointing method does the checkpointing at two different intervals of time steps. The wider interval of the two is called the coarse checkpointing interval and the smaller one is called the fine checkpointing interval. In this text I will refer to the coarse checkpointing interval as n_c and fine checkpointing interval as n_f respectively. The illustration of a two level checkpointing is shown in Figure 6.3, where the red horizontal bars represent coarse checkpoints and the blue horizontal bars represent fine checkpoints. At any point in the simulation, all the coarse checkpoints (red)

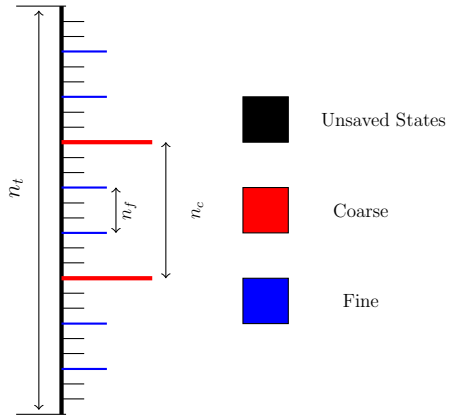


Figure 6.3: The vertical line represents the time scale from 0 to n_t in a two-level checkpointing scheme. Thin black horizontal lines show the states whuch are not checkpointed. Coarse checkpoints are shown in red horizontal lines while fine checkpoints are shown in blue lines. All coarse checkpoints and fine checkpoints between 2 consecutive coarse checkpoints need to be stored.

and one set of fine checkpoints between two consecutive coarse checkpoints are saved. In my implementation, I save the coarse checkpoints in NVMe/HDD and the fine checkpoints are stored in the system RAM.

Two-level checkpointing allows time reversal through three stages.

1. Forward Modeling
2. Repopulating Finer checkpoints
3. Reconstruction

This is shown in Figure 6.4 for a 2nd order ($n_o = 2$) in time adjoint problem with values of $n_t = 20$, $n_f = 5$ and $n_c = 10$. The first pass consists of forward evaluations from $t = 0$ to $t = n_t$. During this pass, all coarse checkpoints

(red squares) are stored in NVMe and all fine checkpoints since the last coarse checkpoints (blue squares) are stored in RAM. The 2 reconstruction sweeps re-constructs all states (gray squares) between the fine checkpoints. Next, the re-population step consists of forward evaluations from $n_t - 2n_c$ to $n_t - n_c$ that repopulate the fine checkpoints (blue). The re-construction sweeps are then repeated which reconstructs all intermediate states(gray squares) using the finer checkpoints(blue). This process is repeated until all states are re-constructed and cross-correlated from $t = n_t$ to t=0. The adjoint states are computed in order of their access so they do not need to be checkpointed.

6.3.2 Compute model

Assume an adjoint workflow with n_t time steps with a time-stepping order of n_o, the fine checkpointing interval, n_f, and the coarse checkpointing interval n_c. I compute the number of time stepping kernel evaluations given any value of n_f, n_t and n_c. During the forward phase, the number of time stepping evaluations, N_1 is given by

$$N_1 = n_t. \tag{6.5}$$

The repopulating and reconstruction phases are where the choice of checkpoint frequency affects the balance of compute and data movement. In the re-populating state, the fine checkpoints between two coarse checkpoints need to be saved. Therefore, I need another $n_c - n_o$ forward evaluations repeated $\frac{n_t}{n_c} - 1$ (Last coarse stride does not need to be repopulated) times to

save the fine checkpoints. Therefore the number of forward evaluations in the re-population phase, N_2, is given by

$$N_2 = (n_c - n_o)\left(\frac{n_t}{n_c} - 1\right), \tag{6.6}$$

because $n_c \gg n_o$ I approximate equation 6.6 as

$$N_2 = n_t - n_c. \tag{6.7}$$

During the reconstruction phase one needs to evaluate $n_f - n_o$ kernels $\frac{n_c}{n_f}$ times every coarse checkpoint interval across a total of $\frac{n_t}{n_c}$ coarse checkpoint intervals. Therefore the total number of computes for the reconstruction phase, N_3, is given by

$$N_3 = (n_f - n_o)\frac{n_c}{n_f}\frac{n_t}{n_c}. \tag{6.8}$$

To compute the total number of computes, I add equations 6.5, 6.7 and 6.8 to obtain

$$N_{forward} = n_t + \frac{n_f - n_o}{n_f} n_t + (n_t - n_c), \tag{6.9}$$

where $N_{forward}$ is the total number of forward timestepping required to compute the solution. Equation 6.9 describes the number of timestep evaluations and ignores any extra data movement arising from loading/saving checkpoints. Analyzing the second term in equation 6.9, one can notice that keeping n_t and

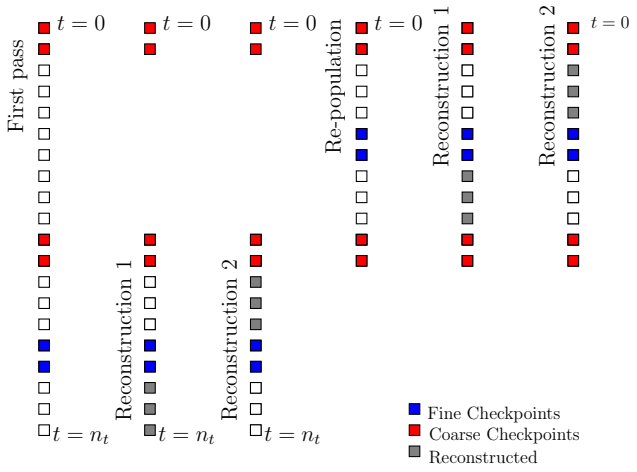


Figure 6.4: Figure showing time reversal using a two-level checkpointing approach with $n_f = 5$ and $n_c = 10$ for a second-order in time adjoint scheme. The first foward sweep stores the coarse checkpoints and the fine checkpoints after the last coarse checkpoint while the next two reconstruction passes reconstruct the intermediate states shown in gray squares. The next pass repopulates the fine checkpoints before the last coarse checkpoint and the final 2 passes reconstruct from each of the fine checkpoints. Imagine many more time steps and only being able to fit the gray states in GPU memory, the blue states in system RAM, and the red states in a slower storage area.

n_c constant, the number of forward evaluations increases as the interval between fine checkpoints, n_f, increases. In other words, one can trade more frequent fine checkpointing for a lower number of kernel evaluations.

The adjoint sweep is not checkpointed and is given by $N_{adjoint}$

$$N_{adjoint} = n_t. \tag{6.10}$$

6.4 Asynchronous Implementation

I hide all data movement arising form loading/store of checkpoints by overlapping it with compute using an asynchronous implementation. I describe the asynchronous implementation in two stages. The first stage covers the forward and the re-population phase while the second one covers the adjoint and re-construction phase.

6.4.1 Forward and re-population stages

During the forward phase, I hide the data transfer from GPU to RAM behind n_f computations. I allocate $n_f + n_o$ buffers and write each snapshot to the $mod(t, n_f + n_o)^{th}$ buffer where $mod()$ is the modulus operator. This is done so that the two buffers that are being transferred do not get over-written by kernel evaluations. If the two states just transferred in RAM correspond to a coarse checkpoint, they are asynchronously written to NVMe until the next coarse checkpoint is evaluated. Therefore, the states saved to NVMe is effectively hidden behind $\frac{n_c - n_f}{n_f}$ fine checkpoints. The repopulation phase works in exactly the same way except that NVMe write is not performed and only the fine checkpoints are stored in RAM. The pictorial representation of the phase is shown in Figure 6.5. Additionally, the pseudo-code for the forward and re-population phases can be seen in Algorithms 1 and 2.

Algorithm 1 Forward propagation psuedo-code describing n_t time steps, S(t) represents state at time t while C and F represents the coarse and fine checkpoints in RAM respectively. f is the function which advances the problem forward in time.

```
1: for t_c = 1 : n_c : n_t do
2:     for t_f = t_c : n_f : t_c + n_c do
3:         Begin Sync
4:         for t = t_f : t_f + n_f do
5:             S(t) = f(S(t − 1), S(t − 2))                    ▷ Stream1
6:         end for
7:         if coarse checkpoint then
8:             C ← S(t − 1) & S(t − 2)                         ▷ Stream2
9:             NVME ← C                                        ▷ AIO Thread
10:        else if active fine checkpoint then
11:            F ← S(t − 1) & S(t − 2)                         ▷ Stream2
12:        end if
13:        End Sync
14:    end for
15:    while NVME write do
16:        wait()
17:    end while
18: end for
```

6.4.2 Adjoint and reconstruction stages

The adjoint and reconstruction phases work in a slightly different way. Before starting the adjoint stage the last n_f forward states are already in memory so one does not need to recompute them from a checkpoint. During the time n_f adjoint states are computed, the previous fine checkpoint is loaded into two buffers where the n_f forward buffers are not present. In the next sweep $n_f - n_o$ evaluations are performed to fill in n_f buffers with reconstructed states. Then adjoint states and related computations are performed while asynchronously loading the previous coarse checkpoint from NVMe/HDD. This is shown in Figure 6.6 and the psuedo-code is presented in algorithm 3.

Algorithm 2 Re-population pseudo-code describing n_t time steps, S(t) represents state at time t while F represents fine checkpoints in RAM. f is the function which advances the problem forward in time.

```
1: for t_f = t_c − n_c + 1 : n_f : t_c do
2:     Begin Sync
3:     for t = t_f : t_f + n_f do
4:         S(t) = f(S(t − 1), S(t − 2))                    ▷ Stream1
5:     end for
6:     if  active fine checkpoint then
7:         F ← S(t − 1) & S(t − 2)                         ▷ Stream2
8:     end if
9:     End Sync
10: end for
```

Equation 6.9 is directly proportional to total compute time if one is able to hide all data movement asynchronously. I now explore checkpointing parameters that can help with this goal.

Algorithm 3 Re-construction & adjoint psuedo-code describing n_t time steps, S(t) and S'(t) represents forward and adjoint states at time t while F represents fine checkpoints in RAM. f is the function which advances the problem forward in time while f' solves the adjoint problem. g is the function that computes with the forward and adjoint states at time t)

1: **for** $t_c = n_t : -n_c : 1$ **do**
2: **if** $t_c \neq n_t$ **then**
3: Repopulate using algorithm 2
4: **end if**
5: $S(t_c - 2n_c) \& S(t_c - 2n_c + 1) \leftarrow NVMe$
6: **for** $t_f = t_c : -n_f : t_c - n_c$ **do**
7: **for** $t = t_f - n_f + n : 1 : t_f$ **do**
8: $S(t) = f(S(t-1), S(t-2))$ ▷ Stream1
9: **end for**
10: **Begin Sync**
11: **for** $t = t_f : -1 : t_f - n_f$ **do**
12: $S'(t) = f'(S'(t+2), S'(t+1))$ ▷ Stream1
13: $g(S(t), S'(t))$ ▷ Stream1
14: **end for**
15: **if** `active fine checkpoint` **then**
16: $S(t_f - 2n_f)$ & $S(t_f - 2n_f + 1) \leftarrow F$ ▷ Stream2
17: **else if** `coarse checkpoint` **then**
18: $S(t_c - n_c)$ & $S(t_c - n_c + 1) \leftarrow C$ ▷ Stream2
19: **end if**
20: **End Sync**
21: **end for**
22: **while** NVME read **do**
23: wait()
24: **end while**
25: **end for**

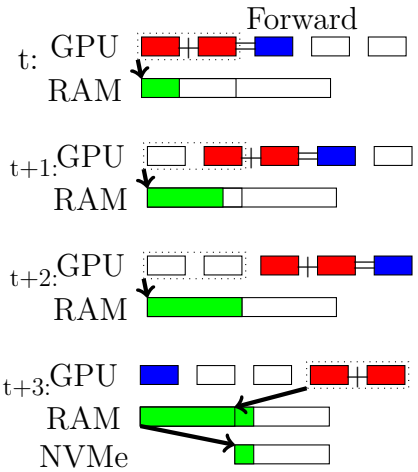


Figure 6.5: Data movement during the forward and re-population phases of the two-level checkpointing for a second order in time adjoint problem with $n_f = 3$ across 4 time-steps. Red rectangles are the input states while blue is the output state. The dotted rectangle encompasses the buffers currently being transferred to the System RAM and the arrows denote the direction of data movement. The buffers in RAM are shown in white and green refers how much data has been transmitted. From $time = t$ to $time = t + 2$, the two buffers are being transferred into system RAM. Once the transfer is done, the same buffer is written into NVMe until the next coarse checkpoint overwrites it. The subsequent finer checkpoints are stored in other RAM buffers.

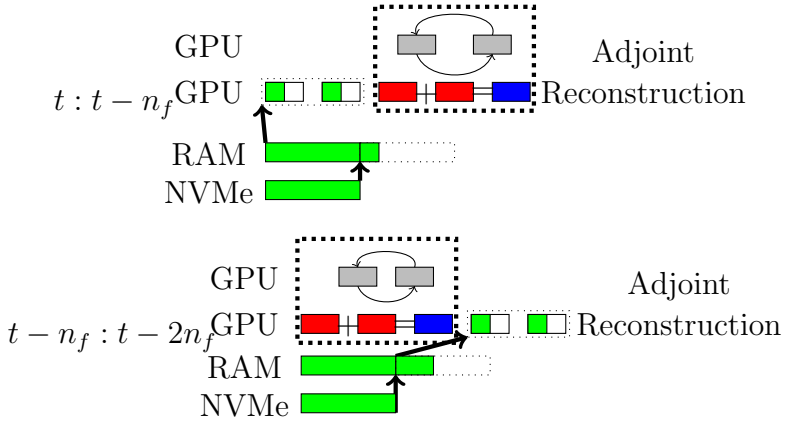


Figure 6.6: Data Movement during the reconstruction and adjoint phases with $n_f = 3$ for a second order in time adjoint problem. The two adjoint buffers are shown in grey and the circular arrows represents each buffer storing every alternate time-steps Forward and Reconstruction phases of the two-level checkpointing based FWI engine with $n_f = 3$. Red rectangles are the input states while blue is the output states. During evaluation of n_f time steps of adjoint 2 buffers are transferred to the GPU buffers enclosed by the thin black dotted rectangle. The adjoint states are computed with the remaining n_f buffers where forward states are already pre-computed. Thick dotted rectangle shows the collection of Forward and adjoint states at the same times. Similar to the forward propagation, data is read from NVMe while n_c kernel evaluations take place asynchronously.

6.4.3 Strategies for selecting optimal checkpointing parameters

I outline how to select optimum checkpoint intervals from system parameters and compute times. Because copies between CPU and GPU are hidden under kernel compute one would want to finish copying one checkpoint **S** within n_f kernel evaluations. Therefore the value of optimal n_f is given by

$$n_f^* = \frac{T_{D2H}}{T_K}, \tag{6.11}$$

where T_{D2H} is the time required to copy one checkpoint from GPU to RAM and T_K is the time required to compute one time step. Equation 6.11 suggests that lower value of T_{D2H} or higher transfer bandwidth means lower value of n_f and in turn lesser GPU memory usage.

I also want to hide the transfer between CPU and NVMe between a multiple of fine checkpointing intervals n_f. Therefore, the optimal ratio of $\frac{n_c}{n_f}$ will be given by

$$\frac{n_c^*}{n_f} = \frac{T_R}{T_{D2H}} + 1, \tag{6.12}$$

where T_R is the time to write one checkpoint to NVMe/HDD. Equation 6.12 signifies that lower value of T_R or higher IO bandwidth means lower value of n_c and in turn less RAM usage. Because the amount of data transferred from GPU to RAM is the same as that transferred from RAM to NVMe/HDD, equation 6.12 can be recast as a ratio of bandwidths given by

$$\frac{n_c^*}{n_f} = \frac{\beta_{D2H}}{\beta_W} + 1, \tag{6.13}$$

where β_{D2H} is the Device-to-Host bandwidth and β_W is the write speed to NVMe/HDD. In practice, both n_f and n_c as obtained by equation 6.11 and 6.12 are real numbers which need to converted to the nearest integer to apply in an adjoint problem.

6.4.4 Memory Constraints

Asynchronous execution typically requires more memory compared to its synchronous counterpart. I outline the memory requirements for the proposed two-level checkpointing method. When I attempt to asynchronously hide the Device-to-Host copy of the wavefield under n_f wave-advance kernels, I need $n_f + n_o$ buffers to store all forward states. Since adjoint states need not be checkpointed I need only n_o adjoint states. Therefore, the total number of wavefields at the lowest level will be $n_f + 2n_o$. As n_f increases, the memory requirement also increases and the GPU memory constraint is

$$(n_f + 2n_o)\ sizeof(P) \leq M_{GPU}, \tag{6.14}$$

where M_{GPU} is the available global GPU memory and P refers to one state buffer.

In the same way, the fine checkpoints between two coarse checkpoints are stored in RAM. Therefore, the number of checkpoints to store in RAM is $\frac{n_c}{n_f}$. Therefore, the system memory constraint is given by

$$\frac{n_c}{n_f}\ sizeof(\mathbf{S}) \leq M_{CPU}, \tag{6.15}$$

where M_{CPU} is the available system RAM.

The coarse checkpoints are stored in NVMe/HDD which needs to hold $\frac{n_t}{n_c}$ checkpoints. Therefore, the NVMe/HDD memory constraint is given by

$$\frac{n_t}{n_c}\ sizeof(\mathbf{S}) \leq M_{NVMe}, \tag{6.16}$$

where M_{NVMe} is the available storage in NVMe/HDD.

6.4.5 Performance model when partial data transfer is hidden

Often, due to memory constraints discussed in section 6.4.4 we need to select $n_f < n_f^*$ which do not hide the data movement completely. equation 6.9 is valid only when $n_f \geq n_f^*$. We now present a simplistic model to model run-times when general values of n_f are used due to memory constraints.

The general model for $n_c < n_c^*$ is complex to quantify during the re-population process because the read from NVMe/HDD is hiding behind n_c adjoint evaluations as well as $n_f - n_o$ forward reconstructions. Also, since the read speed is greater than the write speed, the choice of n_c has a lesser effect during the reconstruction/adjoint phase compared to the forward phase. To keep the model simple, I omit analysis with general n_c and assume $n_c \geq n_c^*$ in this section.

I divide my model for sub-optimal compute in four stages:

1. Forward modeling

2. Re-population

3. Reconstruction

4. Adjoint modeling

In the forward sweep I divide n_t time-steps in segments of n_f time steps giving us a total of $\frac{n_t}{n_f}$ segments. $\frac{n_t}{n_c}$ coarse checkpoints need to be moved from GPU to RAM before writing it to NVMe. Therefore, $\frac{n_t}{n_c}$ segments will overlap with data movement. In the same way, $\frac{n_c}{n_f}$ fine checkpoints between the last two coarse checkpoints need to be moved from GPU to RAM. Therefore, the total number of segments which will overlap with data movement are $\frac{n_t}{n_c} + \frac{n_c}{n_f}$.

Because each segment of n_f kernel evaluations will be waiting until the data is copied between the CPU and GPU each will take T_{D2H} time to complete. Therefore, the total time is given by summing up the contributions of overlapped and non-overlapped segments given by

$$T = \left(\frac{n_t}{n_c} + \frac{n_c}{n_f}\right) * T_{D2H} + \left(\frac{n_t}{n_f} - \frac{n_t}{n_c} - \frac{n_c}{n_f}\right) * n_f * T_K \tag{6.17}$$

In order to simplify equation 6.17 we refer to equation 6.11 and rewrite

$$T_{D2H} \approx n_f^* * T_K \tag{6.18}$$

Effectively, we can write,

$$T = \left[\left(\frac{n_t}{n_c} + \frac{n_c}{n_f}\right) * \frac{n_f^*}{n_f} + \left(\frac{n_t}{n_f} - \frac{n_t}{n_c} - \frac{n_c}{n_f}\right)\right] * n_f * T_K, \tag{6.19}$$

where the term $\frac{n_f^*}{n_f}$ refers to the ratio of optimal value of n_f to the value of n_f in use due to memory constraint. This ratio is capped to 1 as when $n_f > n_f^*$ the $\frac{n_f^*}{n_f}$ term remains 1 as the compute no longer waits for data transfer. We will use this term to estimate effective compute time in both reconstruction and adjoint phases.

In the re-population phase, the total number of kernel evaluations are $n_t - n_c$ or $\frac{n_t - n_c}{n_f}$ segments, out of which every segment is overlapped with moving fine checkpoints from GPU to RAM. Therefore the runtime model for this segment is

$$\left[\frac{n_t - n_c}{n_f} * \frac{n_f^*}{n_f}\right] * n_f * T_K \tag{6.20}$$

The re-construction phase evaluates $n_f - n_o$ kernel evaluations per segment with a total of $\frac{n_t}{n_f}$ segments. None of the segments are overlapped as no data movement happens during the reconstruction phase. The effective time spent during this phase is

$$\left[\frac{n_f - n_o}{n_f} * \frac{n_t}{n_f}\right] * n_f * T_K \tag{6.21}$$

During the adjoint computation, the total number of kernel evaluations is n_t or $\frac{n_t}{n_f}$ segments. All segments in the adjoint sweep are overlapped waiting

for the fine checkpoints to be transferred from RAM to GPU. Therefore, the effective compute is

$$\left[\frac{n_t}{n_f} * \widetilde{\frac{n_f^*}{n_f}}\right] * n_f * T_K^{'}, \tag{6.22}$$

where the ˜ term denotes the value n_f^* computed on the basis of Host-to-Device transfers and $T_K^{'}$ is the time taken to execute one adjoint time stepping.

To simplify equations 6.19 , 6.20, 6.21, and 6.22 let $n_f * T_K = T_f$ and $n_f * T_K^{'} = T_f^{'}$. As a result, equation 6.19 becomes

$$\left[\left(\frac{n_c}{n_f} + \frac{n_t}{n_c}\right)\left(\frac{n_f^*}{n_f} - 1\right) + \frac{n_t}{n_f}\right] * T_f \tag{6.23}$$

Similarly equation 6.20 can be simplified as

$$\left[\left(\frac{n_t}{n_f} - \frac{n_c}{n_f}\right)\left(\frac{n_f^*}{n_f}\right)\right] * T_f \tag{6.24}$$

While equation 6.21 reduces to

$$\frac{n_f - n_o}{n_f} * \frac{n_t}{n_f} * T_f \tag{6.25}$$

and equation 6.22 reduces to

$$\left[\frac{n_t}{n_f}\widetilde{\frac{n_f^*}{n_f}}\right] * T_f^{'} \tag{6.26}$$

Summing up equations 6.23, 6.24, 6.25, and 6.26, we obtain the total theoretical compute time as

$$\left[-\frac{n_c}{n_f}+\frac{n_t}{n_c}\left(\frac{n_f^*}{n_f}-1\right)+\frac{n_t}{n_f}\left(1+\frac{n_f-n_o+n_f^*}{n_f}\right)\right]T_f + \left[\frac{n_t}{n_f}\frac{\widetilde{n_f^*}}{n_f}\right]*T_f^{'} \tag{6.27}$$

An interesting observation is that putting $n_f = n_f^*$ in equation 6.27 gives the same compute as equation 6.9 for forward evaluations and equation 6.10 for adjoint evaluations.

The takeaways from equation 6.27 are the following

- Because the value of $\frac{n_f^*}{n_f}$ has to be ≥ 1, selecting $n_f > n_f^*$ still hides data movement. However, it will increase the global memory usage of the GPU as per equation 6.14
- Using $n_f > n_f^*$ can be a detriment to runtimes in some cases as higher n_f also increases the number of re-computations. This is because when $n_f \geq n_f^*$ equation 6.27 collapses to equation 6.9. The only term affecting amount of compute is the second term in equation 6.9 which increases with increasing n_f. Therefore, $n_f = n_f^*$ is actually an optimal fine checkpointing interval that minimizes runtimes.

6.4.6 Benefits of faster interconnects and high speed on-node memory

From the memory constraints and sub-optimal compute model, It is clear that a faster interconnect gives a lower value of n_f while a faster on-node

storage gives a lower value of n_c. This is an extremely significant observation as it directly increases global memory requirement in the GPU (equation 6.14). Similarly, a higher value of $\frac{n_c}{n_f}$ requires more RAM for storage of all fine checkpoints. As a result, faster interconnects and on-node memory can load/offload data much faster so that it does not create a bottleneck for the higher throughput of GPU. This also means that, for the same amount of GPU and system memory, larger problems can be solved in a pipelined fashion. This usable domain of pipelined execution can be seen in Figure 6.7. The shaded area in blue shows the difference in problem sizes that NVMe+NVLink can handle which the PCIe+HDD combo cannot.

Equation 6.27 outlines that a faster interconnect can have lower values of n_f and n_c which are helpful in reducing wall clock time directly. Therefore, NVLink + NVMe are not only able to execute codes faster but can also do so with bigger problem using the same amount of memory.

6.5 Results

I ran tests on a IBM machine with Power 8 processor with 512 GB of system RAM and 4 Nvidia P100 GPUs with NVLink per node. Each P100 contains 3840 CUDA cores with 16GB of global memory. Additionally, each node had a 1.5TB NVMe. For comparison, I ran the same workflow on another P100 based system where the GPUs were connected through PCIe. The PCIe nodes did not have NVMe so I simulated PCIe results by comparing the data transfers through PCIe and NVLink.

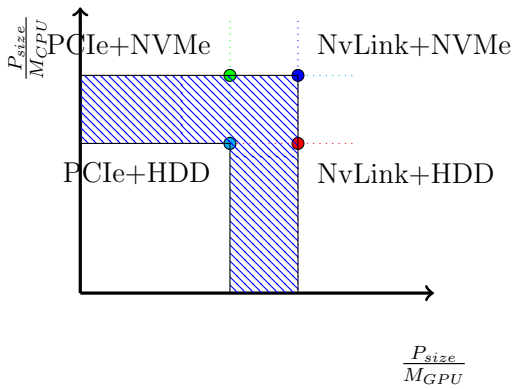


Figure 6.7: Figure showing the usable domain of streaming adjoint problem with different interconnects (NVlink vs PCIe) and storage (HDD vs NVMe) . The two axes show the size of a single checkpoint relative to available memory in the GPU and CPU. From memory constraints inequalities, I conclude that a faster interconnect will have a lower memory requirement in the CPU as well as RAM. The slower interconnects need more memory to compute in a streamlined function otherwise they end up waiting for data movement.

6.5.1 Code set up

I wrote the FWI code in CUDA C version 9.0 and MPI was used to manage different GPUs. Because the code was benchmarked on multiple nodes, I did not use P2P for halo exchanges between GPUs. I copied them to the host and did the exchange using the $MPI_Sendrecv$ function. Although P2P transfers are faster, MPI is imperative to exchange halos between GPUs residing in different nodes. Asynchronous IO is implemented using the AIO library.

6.5.2 Optimal checkpointing values

I ran a velocity model with dimensions of 420x1648x848 points for FWI. I timed the kernel evaluation on each GPU and obtained an average of 11.1ms. Because this is a second order in time implementation, the value of S will be two wavefield arrays. Two states of wavefield took 67.5ms to transfer from Device-to-Host at a throughput of 18.49GB/s per GPU. This bandwidth is less than the NVLink bandwidth of 40GB/s because it is limited by the memory write bandwidth, which is about 40GB/s per socket. Therefore, n_f^* is given by equation 6.11 which comes out to be 6.03. During adjoint computation, the kernel evaluation time was a combination of the adjoint solver and the laplacian and cross-correlation operation which came out to be 12.45ms. However, two wavefield states were loaded to the GPU in 44.39 ms at a throughput of 28.13 GB/s making $\widetilde{n_f^*} = 3.646$. The Host-to-Device throughput is higher because memory read bandwidth is higher for Power 8 CPUs.

For the PCIe based test, I got a similar kernel evaluation time of 11.12ms. However, the time taken to transfer two buffers of wavefield from Device-to-Host took 106ms at a throughput of 12.46GB/s a giving value of $n_f^* = 9.534$. During adjoint computation, the throughput obtained from Host-to-Device transfers was 12.13GB/s, giving $\widetilde{n_f^*} = 9.2799$.

Similarly, following equation 6.12, I calculate n_c^* for NVMe as well as Disk Drives. The NVMe drive I benchmarked attained a write speed of 2.1GB/s while the aggregate D2H bandwidth was 74GBps (18.5GBps per GPU) giving $n_c^* = 36.2$. The gpfs system, on the other hand, obtained a

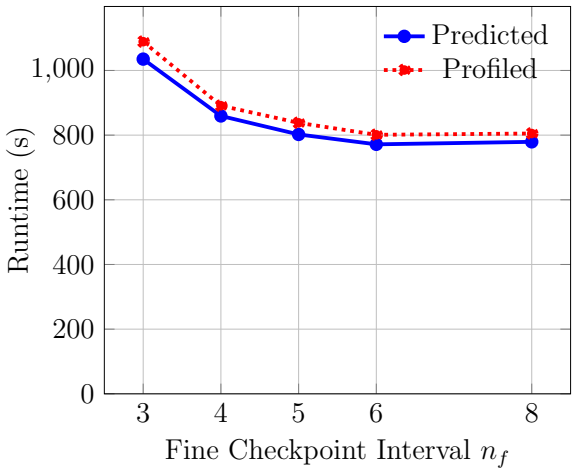


Figure 6.8: Predicted vs observed runtimes for a FWI gradient problem for different values of n_f using NVLink and NVMe. The predicted values were computed using equation 6.27 and appear to be in good agreement with observed values

peak write speed of 240MB/s, giving of $n_c^* = 309.33$.

6.5.3 Runtimes (Predicted vs Computed)

I selected different fine checkpointing intervals while keeping the ratio of n_c and n_f fixed at 80. I ran the FWI engine implementation for 19200 time steps, and show the predicted and observed runtimes in Figure 6.8. One can notice that the predicted and computed times are very close to each other. The time difference between the two can be attributed to synchronization operations using different streams in CUDA C as well as the CPU overhead.

An interesting observation is that the code runs faster with $n_f = 6$(792s) compared to $n_f = 8$(801s) although $n_f = 8$ hides all the compute. This is because the ratio $\frac{\widetilde{n_f^*}}{n_f}$ is always capped at 1 and using a value of $n_f \geq n_f^*$

does not make the code faster. Infact, it makes it slower because it increases the number of re-computations. This can be seen in the second term of equation 6.27 which says that as the checkpointing interval increases, the number of re-computations also increase.

6.5.4 Runtimes for different Configuration

I made several runs with different checkpointing parameters on different cluster configurations. I made four combinations combining PCIe/NVLink with NVMe/HDD, and I show the runtimes in Figure 6.9. As predicted from the theoretical models, the NVLink+NVMe waits the least for data movement and is the fastest. Since I did not have access to PCIe+NVMe nodes, I simulated PCIe runs on Nvidia cards by computing the ratio r of D2H and H2D throughputs for NVLink and PCIe and asking the NVLink based cards to transfer the wavefields r times for both H2D and D2H transfers. Since both PCIe and NVLink GPUs are P100 units, this provides a closest approximation on the NVMe+PCIe connections. As expected, PCIe+NVMe was mostly limited by PCIe bandwidth and was slower than NVLINK+NVME result. In fact, similar runtimes were achieved using $n_f = 3$ for NVLink and $n_f = 6$ for PCIe. This gives memory advantage to NVLink as it can run twice the size of PCIe in a pipelined fashion.

The HDD paired examples were severely bottlenecked by IO performance and performed the worst in the tests.

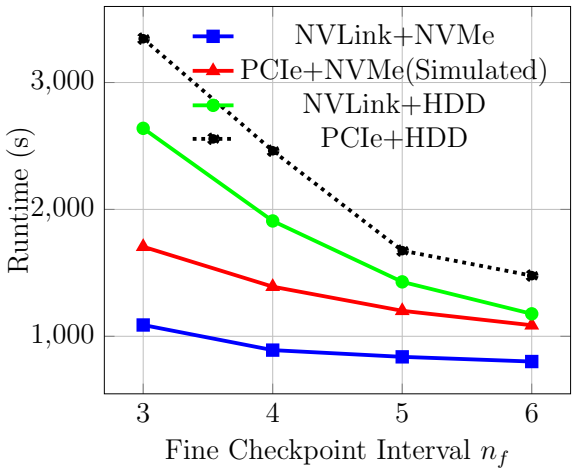


Figure 6.9: Runtimes for a FWI engine for different storage and interconnects. NVLink and NVMe appear to have the best performance while HDD and PCIe are the worst.

6.5.5 Scaling

To compute weak scaling characteristic, I ran the FWI engine implementation with a problem size of 420×848×848 per node. The relative runtimes are shown in Figure 6.10 for different values of n_f. The curves are almost flat, showing perfect weak scaling across nodes.

For strong scaling characteristic, I run a model of size 420×848×1696 in 8, 16 and 32 MPI ranks respectively. The relative runtimes for different configurations are shown in Figure 6.11. The runtimes show that, as the work per processor decreases, the kernel evaluations slow down because halo exchanges over MPI cannot be completely hidden by the compute of internal region.

I also plot a demonstration of the speed-up for different values of n_f,

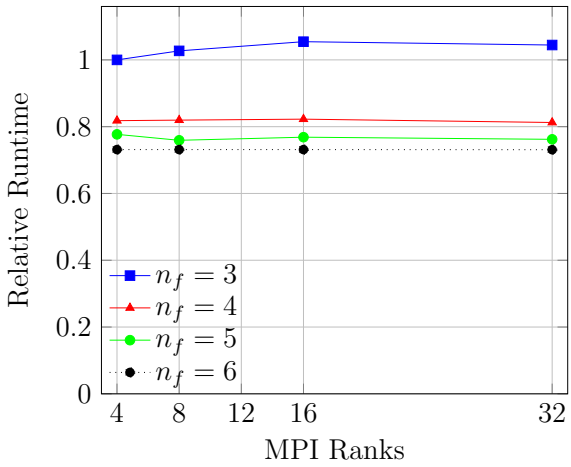


Figure 6.10: Weak Scaling characteristics of the two-level checkpointing approach running with NVLink and NVMe for different values of n_f.

which is shown in Figure 6.12. As seen in the strong scaling results, as the domain size per GPU decreases the MPI overhead takes over compute and linear speed-up is not obtained. With larger models, the speed-up may approach a linear curve.

6.6 Conclusions

I propose a novel asynchronous two-level checkpointing technique which reformulates an adjoint problem as a data movement problem. I present formulations to construct optimal checkpointing parameters for any adjoint problem by modifying values in my models. Not only is the model valid for the current generation of hardware, it can also be modified for the next generation hardware by plugging in the required parameters. The checkpointing scheme is able to minimize the data movement completely when optimal checkpoint-

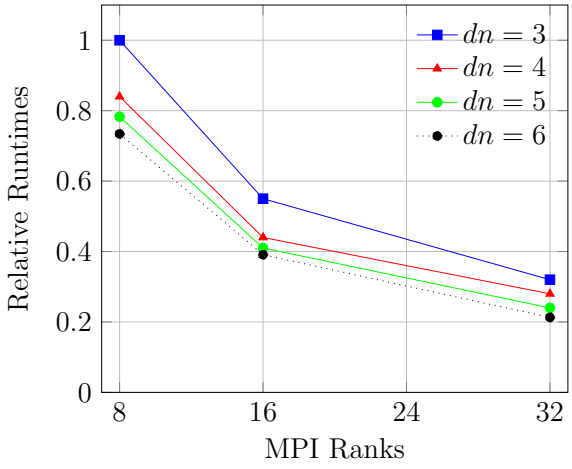


Figure 6.11: Strong Scaling characteristics of the two-level checkpointing approach running with NVLink and NVMe for different values of n_f. The curves suggest that as the work per processor decreases, the MPI overhead to exchange halos is greater than the time to compute the interior domain resulting in imperfect strong scaling.

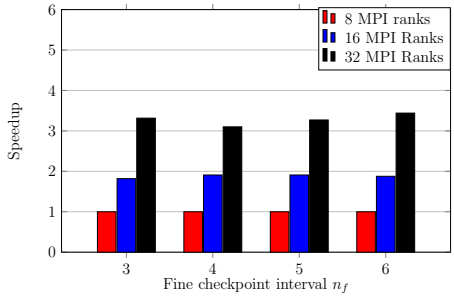


Figure 6.12: Speed-up characteristic for different values of n_f. The speedups are consistent across different values of n_f. Large models may show better speed-up.

ing parameters are used and runtimes can be easily predicted for any value of checkpointing intervals. I observe that fast interconnects are essential for pipelined asynchronous and slower interconnects considerably slow down the RTM/FWI implementation. The two-level checkpointing also achieves very good scaling properties, both strong and weak showing it is a viable method to use in large scale problems. Because this approach is largely limited by data transfer rate in most cases, more expensive and complex kernels would be at an advantage as the values of n_f^* will be lower giving us more memory space to work with a perfectly streamlined implementation.

Chapter 7

Summary

Full waveform inversion has now become a very popular method for estimating velocity models from seismic data in 2D and 3D despite some of its fundamental shortcomings. In my dissertation, I attempted to address two principal issues, namely, estimation of a good starting model for FWI, and fast computation of forward modeling and gradient. A VFSA based algorithm with sparse model parameterization was found to be robust and accurate. Although I was able to obtain good results with PSO, I found that choice of VFSA parameters such as starting temperature and cooling schedule to be much easier and stable than the PSO-gradient parameters. Note also that I avoid the discussion of relative performance of FWI in time and frequency domains. I attempted to speed up forward calculation in the frequency domain and also developed a fast two-step checkpointing algorithm in the time domain. I carried out most of my calculations using GPUs and summarize below the findings of my research.

In part 1 of my dissertation, I presented methods to perform global optimization for FWI using Very Fast Simulated Annealing and Particle Swarm Optimization. The methods have been demonstrated both in 2D and 3D and

they work well for both synthetic and real data when coupled with a suitable sparse parameterization method. The salient features of the proposed methods are as follows:

- Sparse parameterization is essential to reduce the number of model parameters and making global optimization computationally tractable. Three different sparse parameterization techniques have been proposed in this thesis. Any sparse parameterization technique can be used as long it is able to reduce the number of model parameters while maintaining the necessary degrees of freedom.

- A good starting guess is not necessary for the global optimization to converge. One can start anywhere in the search space to converge to nearly similar solutions. I successfully demonstrated the stability of the final model when starting with different starting models within the search space.

- Once the optimization has been set up, no human intervention is required during the runtime of the code. The user only needs to select the search space and the temperature cooling schedule. Thus this is an automated method compared to Migration Velocity Analysis.

- Generally more iterations are required in a global optimization algorithm compared to a local optimization method. However, because the global optimization algorithm is gradient-free, only one forward model evaluation is required per iteration instead of four or more in a gradient FWI.

So even if the number of iterations is more in the methods proposed here, the computational cost is comparable to standard FWI. In addition, no adjoint calculation is needed and therefore, the memory requirement is also reduced.

In part 2, I proposed a method to reduce the computational cost of the modeling in frequency domain and adjoint calculations in the time domain. While the time domain achieves the target by an optimal checkpointing technique, the frequency domain approach exploits constant velocity zones to use its pre-computed response in all subsequent iterations of FWI. The main features of the approaches can be listed as follows:

- The frequency domain Schur complement method exploits unchanging zones to pre-compute its response once and re-use it in every subsequent modeling operation in FWI. When coupled with a suitable global optimization method, it can further optimize runtimes for the the global optimization workflow.

- The two-level checkpointing method is a proposed framework to accelerate general adjoint problems on GPUs using optimal two-level checkpointing parameters. This framework can be extended to any variants (acoustic, anisotropic, elastic) of RTM in FWI. It can also be used in other fields where adjoint problems are essential such as Computational Fluid Dynamics (CFD).

7.1 Future Work

- I propose a two-step approach where we first generate a starting model using global optimization method and then use it as a starting model in Local FWI. One can also combine both approaches where a global solution can be refined at every iteration using gradient-based approaches. Although I have not implemented this approach. I believe that the resolution of inversion can be improved using this technique.

- I present global optimization approaches that can be termed as the first version of an automated inversion algorithm. Recently, Machine Learning has been gaining traction, and the derivative-free framework I propose can serve as a good starting point to implement machine learning in FWI especially in light of a fast modeling method such as the Schur complement.

- I demonstrate the global optimization methods for primarily acoustic data. However, it can be easily extended to elastic and anisotropic formulations. The framework will remain the same and an alternate sparse parameterization technique may be required. With multi-parameter estimation, the algorithm may take more iterations to converge but that is the only change I can foresee in the extension of the approach. The issue of cross-talk between different parameters needs to be addressed as well.

- The two-level checkpointing method is future-proof by design and any

new platform can have an efficient checkpointing algorithm using this approach. Unless radical changes are made in the hardware design, the concept of asynchronous processing can be used to reduce runtimes of several different applications.

Appendices

Appendix A

Swarm optimization in FWI

A.0.1 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a swarm intelligence optimization technique proposed by Kennedy (2010). It is inspired by the phenomenon of bird flocking where a group of birds (models in this case) communicate among themselves to find the location of the food (global minimum in this case). We imitate the heuristics of birds in an optimization problem to reach the global minimum.

PSO initializes a population of n particles/models $[\mathbf{m}_0, \mathbf{m}_1 ..... \mathbf{m}_n]$ initialized randomly across the search space. At each iteration we define two values $\mathbf{m}_i^{best}$ for each particle and a single $\mathbf{g}^{best}$. $\mathbf{m}_i^{best}$ defines the best position attained so far by each particle while $\mathbf{g}^{best}$ is the best position obtained by any particle until now.

At each iteration, each particle is updated according to a derived update vector given by

$$\mathbf{u}_i^{k+1} = \mathbf{u}_i^k + c1 \cdot rand() \cdot \left(\mathbf{m_i^{best}} - \mathbf{m}_i^k\right) + c2 \cdot rand() \cdot \left(\mathbf{g^{best}} - \mathbf{m}_i^k\right), \quad \text{(A.1)}$$

where $\mathbf{u}_i^k$ and $\mathbf{m}_i^k$ are the update vector and position of ith particle at kth iteration respectively. $c1$,$c2$ are empirical constants and $rand()$ refers to a random number in $(0, 1)$. We initialize $\mathbf{u}_i^0$ as a zero vector.

The tuning parameters $c1$ and $c2$ are normally selected in $(0, 3)$ and are selected by trail and error. The model is iteratively updated as

$$\mathbf{m}_i^{k+1} = \mathbf{m}_i^k + a.\mathbf{u}_i^k, \tag{A.2}$$

where γ is the stochastic step length fixed at 0.1.

Update vectors for each dimension each dimension are clamped to a maximum velocity $\mathbf{u}_{max}$. If the magnitude of u_i^k exceeds $\mathbf{u_{max}}$ then u_i^k is set to $\mathbf{u_{max}}$, which is a parameter specified by the user.

A.0.2 PSO gradient hybrid optimization

The basic motivation of combining the gradient and PSO methods is that the PSO part helps models escape the local minima while the gradient method helps solutions guide in the direction of decreasing misfit. Starting with gradient iterations brings each model near its corresponding local minimum well. From there the positions of the model are modified as per PSO rule which slowly enables a few models in the population to reach the global minima well.

We combine the two algorithms the following way. We perform a fixed set of gradient iteration between every PSO iteration. From equation A.1 we observe that the update to velocities is random. To better constrain the

model updates, we modify the first term of equation A.1 by replacing it with the gradient derived from the adjoint state method following equation 1.4. The algorithm of our implementation is described below.

Algorithm 4 Hybrid PSO Pseudo code: Consider n PSO iterations with k gradient iteration. The number of particles in PSO is n and G represents the data misfit function from equation 1.3 with gradient computed from Equation 1.4.

for $k = 1 : niter$ **do**
  for $i = 1 : nparticles$ **do**
    for $k = 1 : graditer$ **do** ▷ Gradient FWI iterations
      $E = G(m_i^k)$ ▷ Equation 1.3
      $\nabla E = \frac{\partial E}{\partial m_i^k}$ ▷ Equation 1.4
      $m_i^k = m_i^k + \alpha \nabla E$
    end for
  end for
  Find m_i^{best} *and* g^{best}
  for $i = 1 : nparticles$ **do** ▷ PSO iterations
    $\mathbf{v}_i^{k+1} = \nabla E + c1 \cdot rand() \cdot \left(\mathbf{m_i^{best}} - \mathbf{m}_i^k\right) + c2 \cdot rand() \cdot \left(\mathbf{g^{best}} - \mathbf{m}_i^k\right)$
    $\mathbf{m}_i^{k+1} = \mathbf{m}_i^k + \gamma \mathbf{m}_i^k$
  end for
end for

A.1 Results

We apply our proposed method to the SEG-EAGE overthrust model shown in Chapter 2 in Figure 2.5a. Similar to the approach in chapter 2, I select a few events in the zero offset seismic section shown in Figure 2.5b and draw random models from the velocity-interface parameterization. The data was generated with a 5Hz wavelet and the data was inverted in one go like the VFSA example. One of the random starting models is shown in Figure A.1. We

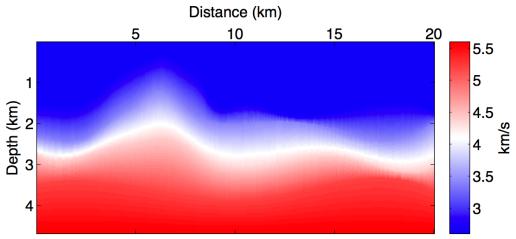


Figure A.1: A random starting model generated from the picked interfaces using the sparse parameterization technique after Datta and Sen (2016)

run hybrid PSO for 10 iterations with 20 gradient iterations per PSO iteration with a population size of 10. The final velocity model after 10 iterations of PSO is shown in Figure A.2 . The evolution of error vs forward evaluations for the best final model can be seen in Figure A.3. The blue lines in the figure display the error evolution during the gradient iterations whereas the red lines are the error steps taken by the PSO update rules. We see that the PSO steps sometimes improves the solution and sometimes worsens it. This allows it to escape a local minimum otherwise not possible with the conventional FWI.

One drawback of the PSO approach is the extremely large number of model evaluation compared to VFSA. Figure A.3 shows the number of forward evaluations per particle to be about 250 which is much higher than what VFSA requires.

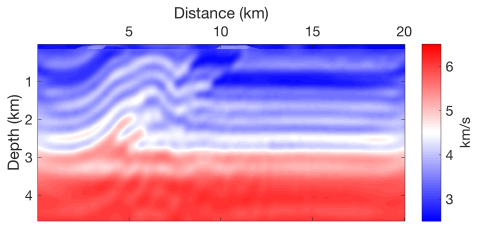


Figure A.2: The final inverted best solution for the Overthrust model after 10 iterations of hybrid PSO-FWI

A.2 Discussions and conclusions

A.2.1 PSO tuning factors

PSO requires careful selection of three tuning factors a, b and c, While the selection of a (constriction factor) is trivial, the factors b and c are not. Kennedy (2010) suggested values of a and b to be 2.8 and 1.3 respectively based on empirical tests. However, our tests on FWI show that we could obtain optimal convergence at values much less than those. Moreover, the values change with the problem. So it may be required to carry out a small sensitivity test before undertaking inversion of a dataset to select optimal values of b and c.

A.2.2 Population Size and iterations

The trade-off between population size and the number of iterations is purely dictated by the computational cost. While the simpler overthrust model

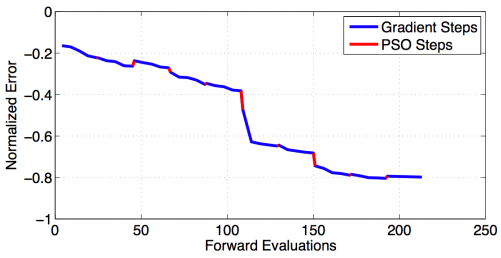


Figure A.3: The error vs forward evaluations plot for the best model for the Overthrust model. The blue lines show the gradient steps while the red curves show the PSO updates

converged using only 10 particles, more complex model can require a higher population size.

A.2.3 Quality of the starting model

The quality of the starting model does not generally affect convergence of the hybrid algorithm but a well-constrained starting model helps in faster convergence in a finite number of iterations thereby reducing the computation cost for the inversion. The starting model we use here has been derived from a few interfaces. However, a heavily smoothed models with different smoothing radius can also be used as starting models.

A.3 Conclusions

We have demonstrated a hybrid FWI formulation combining both gradient and PSO updates. The gradient iteration helps drive the solution towards local minima while the PSO iterations help it escape local minima. Using only

10 particles, we are able to obtain satisfactory convergence very close to the true model. This approach takes more iterations to converge that the standard gradient-based approach but the added benefit of this method is that it does not require a good starting model in order to converge.

However, PSO is less attractive compared to VFSA because of its computational cost. Even using 10 particles increases the computational cost by a great deal. Generally, the selection of tuning factors from PSO is extremely difficult. We prefer the ease of tuning as well as the lower computational cost of VFSA.

Bibliography

Afanasiev, M. V., R. G. Pratt, R. Kamei, and G. McDowell, 2014, Waveform-based simulated annealing of crosshole transmission data: a semi-global method for estimating seismic anisotropy: Geophysical Journal International, **199**, 1586–1607.

Aleardi, M., and A. Mazzotti, 2017, 1d elastic full-waveform inversion and uncertainty estimation by means of a hybrid genetic algorithm–gibbs sampler approach: Geophysical Prospecting, **65**, 64–85.

Almomin, A., B. Biondi, et al., 2012, Tomographic full waveform inversion: Practical and computationally feasible approach: 82nd Annual International Meeting, SEG, Expanded Abstracts, 1–5.

Amestoy, P. R., I. S. Duff, J.-Y. LExcellent, and J. Koster, 2000, Mumps: a general purpose distributed memory sparse solver: International Workshop on Applied Parallel Computing, Springer, 121–130.

Aminzadeh, F., J. Brac, and T. Kunz, 1997a, 3d salt and overthrust model. modeling series i: Society of Exploration Geophysicists.

Aminzadeh, F., B. Jean, and T. Kunz, 1997b, 3-d salt and overthrust models: Society of Exploration Geophysicists.

Arnulf, A. F., A. J. Harding, S. C. Singh, G. Kent, and W. C. Crawford, 2011, Layer 2A controversy resolved using full waveform inversion of downward

continued seismic reflection data at the Lucky Strike Volcano, Mid-Atlantic Ridge: AGU Fall Meeting Abstracts.

Asaeedi, S., F. Didehvar, and A. Mohades, 2017, α-concave hull, a generalization of convex hull: Theoretical Computer Science.

Asnaashari, A., R. Brossier, S. Garambois, F. Audebert, P. Thore, and J. Virieux, 2013a, Regularized seismic full waveform inversion with prior model information: Geophysics, **78**, R25–R36.

———, 2013b, Regularized seismic full waveform inversion with prior model information: Geophysics, **78**, R25–R36.

Aupy, G., J. Herrmann, P. Hovland, and Y. Robert, 2016, Optimal multistage algorithm for adjoint computation: SIAM Journal on Scientific Computing, **38**, C232–C255.

Bai, J., D. Yingst, et al., 2014, Simultaneous inversion of velocity and density in time-domain full waveform inversion: Presented at the 2014 SEG Annual Meeting, Society of Exploration Geophysicists.

Bautista-Gomez, L., S. Tsuboi, D. Komatitsch, F. Cappello, N. Maruyama, and S. Matsuoka, 2011, Fti: high performance fault tolerance interface for hybrid systems: Proceedings of 2011 international conference for high performance computing, networking, storage and analysis, ACM, 32.

Baysal, E., D. D. Kosloff, and J. W. Sherwood, 1983, Reverse time migration: Geophysics, **48**, 1514–1524.

Ben-Hadj-Ali, H., S. Operto, and J. Virieux, 2011, An efficient frequency-domain full waveform inversion method using simultaneous encoded sources:

Geophysics.

Bharadwaj, P., W. Mulder, and G. Drijkoningen, 2016, Full waveform inversion with an auxiliary bump functional: Geophysical Journal International, **206**, 1076–1092.

Billette, F., and S. Brandsberg-Dahl, 2005, The 2004 bp velocity benchmark: Presented at the 67th EAGE Conference & Exhibition.

Billette, F., and G. Lambaré, 1998, Velocity macro-model estimation from seismic reflection data by stereotomography: Geophysical Journal International, **135**, 671–690.

Brossier, R., S. Operto, and J. Virieux, 2009, Robust elastic frequency-domain full-waveform inversion using the l1 norm: Geophysical Research Letters, **36**.

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

Buono, D., F. Artico, F. Checconi, J. W. Choi, X. Que, and L. Schneidenbach, 2017, Data analytics with nvlink: An spmv case study: Proceedings of the Computing Frontiers Conference, ACM, 89–96.

Capdeville, Y., Y. Gung, and B. Romanowicz, 2005, Towards global earth tomography using the spectral element method: a technique based on source stacking: Geophysical Journal International, **162**, 541–554.

Červený, V., 2000, Seismic ray method: Cambridge University Press.

Chavent, G., and C. A. Jacewitz, 1995, Determination of background velocities by multiple migration fitting: Geophysics, **60**, 476–490.

Choi, Y., D.-J. Min, and C. Shin, 2008, Two-dimensional waveform inversion of multi-component data in acoustic-elastic coupled media: Geophysical prospecting, **56**, 863–881.

Claerbout, J. F., 1971, Toward a unified theory of reflector mapping: Geophysics, **36**, 467–481.

———, 1985, Fundamentals of geophysical data processing.

Claerbout, J. F., and S. M. Doherty, 1972, Downward continuation of moveout-corrected seismograms: Geophysics, **37**, 741–768.

Clapp, R. G., 2009, Reverse time migration with random boundaries, *in* SEG Technical Program Expanded Abstracts 2009: Society of Exploration Geophysicists, 2809–2813.

Crase, E., A. Pica, M. Noble, J. McDonald, and A. Tarantola, 1990, Robust elastic nonlinear waveform inversion: Application to real data: Geophysics, **55**, 527–538.

Dahlke, T., B. Biondi, and R. Clapp, 2015, Domain decomposition of level set updates for salt segmentation, *in* SEG Technical Program Expanded Abstracts 2015: Society of Exploration Geophysicists, 1366–1371.

Dai, W., P. Fowler, and G. T. Schuster, 2012, Multi-source least-squares reverse time migration: Geophysical Prospecting, **60**, 681–695.

Danecek, P., and G. Seriani, 2008, An efficient parallel chebyshev pseudo-spectral method for large scale 3d seismic forward modelling: Presented at the 70th EAGE Conference & Exhibition.

Datta, D., M. Sen, F. Liu, and S. Morton, 2016, Salt model building by shape-

based parameterization and global FWI, *in* SEG Technical Program Expanded Abstracts 2016: Society of Exploration Geophysicists, 1069–1073.

Datta, D., and M. K. Sen, 2016, Estimating a starting model for full-waveform inversion using a global optimization method: Geophysics.

De Basabe, J. D., and M. K. Sen, 2007, Grid dispersion and stability criteria of some common finite-element methods for acoustic and elastic wave equations: Geophysics, **72**, T81–T95.

Dessa, J.-X., S. Operto, S. Kodaira, A. Nakanishi, G. Pascal, J. Virieux, and Y. Kaneda, 2004, Multiscale seismic imaging of the eastern nankai trough by full waveform inversion: Geophysical Research Letters, **31**.

Díaz, E., and A. Guitton, 2011, Fast full waveform inversion with random shot decimation, *in* SEG Technical Program Expanded Abstracts 2011: Society of Exploration Geophysicists, 2804–2808.

Djikpéssé, H. A., and A. Tarantola, 1999, Multiparameter l1 norm waveform fitting: Interpretation of gulf of mexico reflection seismograms: Geophysics, **64**, 1023–1035.

Docherty, P., R. Silva, S. Singh, Z.-M. Song, and M. Wood, 1997, Migration velocity analysis using a genetic algorithm: Geophysical prospecting, **45**, 865–878.

Dorigo, M., and M. Birattari, 2010, Ant colony optimization, *in* Encyclopedia of Machine Learning: Springer, 36–39.

dos Santos, A. W., and R. C. Pestana, 2015, Time-domain multiscale full-waveform inversion using the rapid expansion method and efficient step-

length estimation: Geophysics, **80**, R203–R216.

dos Santos, A. W. G., and R. C. Pestana, 2013, 59, *in* Time-domain multiscale full waveform inversion using the rapid expansion method: 279–284.

Edelsbrunner, H., D. Kirkpatrick, and R. Seidel, 1983, On the shape of a set of points in the plane: IEEE Transactions on information theory, **29**, 551–559.

Fichtner, A., J. Trampert, P. Cupillard, E. Saygin, T. Taymaz, Y. Capdeville, and A. Villaseñor, 2013, Multiscale full waveform inversion: Geophysical Journal International, ggt118.

Fleury, C., and F. Perrone, 2012, Bi-objective optimization for the inversion of seismic reflection data: Combined fwi and mva, *in* SEG Technical Program Expanded Abstracts 2012: Society of Exploration Geophysicists, 1–6.

Forgues, E., 1996, Inversion linearisée multi-paramètres via la théorie des rais: PhD thesis, Institut Français du Pétrole - University Paris VII.

Forgues, E., and G. Lambaré, 1997, Parameterization study for acoustic and elastic ray plus born inversion: Journal of Seismic Exploration, **6**, 253–277.

Gardner, G., L. Gardner, and A. Gregory, 1974, Formation velocity and density-the diagnostic basics for stratigraphic traps: Geophysics, **39**, 770–780.

Gauthier, O., J. Virieux, and A. Tarantola, 1986, Two-dimensional nonlinear inversion of seismic waveforms: Numerical results: Geophysics, **51**, 1387–1403.

Gazdag, J., 1978, Wave equation migration with the phase-shift method: Geophysics, **43**, 1342–1351.

Ghosh, S., T. Liao, H. Calandra, and B. M. Chapman, 2014, Performance of cpu/gpu compiler directives on iso/tti kernels: Computing, **96**, 1149–1162.

Godwin, J., and P. Sava, 2011, A comparison of shot-encoding schemes for wave-equation migration, *in* SEG Technical Program Expanded Abstracts 2011: Society of Exploration Geophysicists, 32–36.

Griewank, A., 1992, Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation: Optimization Methods and software, **1**, 35–54.

Griewank, A., and A. Walther, 2000, Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation: ACM Transactions on Mathematical Software (TOMS), **26**, 19–45.

Guitton, A., and E. Díaz, 2012, Attenuating crosstalk noise with simultaneous source full waveform inversion: Geophysical Prospecting, **60**, 759–768.

Guitton, A., and W. W. Symes, 2003, Robust inversion of seismic data using the huber norm: Geophysics, **68**, 1310–1319.

Gutenberg, B., 1951, Internal constitution of the earth: Dover Publications Inc., **7**.

Ha, W., and C. Shin, 2012, Efficient full waveform inversion using a cyclic shot sampling method, *in* SEG Technical Program Expanded Abstracts 2012: Society of Exploration Geophysicists, 1–5.

Hale, D., 2009, Image-guided blended neighbor interpolation of scattered data.

Holland, J. H., 1992, Genetic algorithms: Scientific american, **267**, 66–72.

Hustedt, B., S. Operto, and J. Virieux, 2004, Mixed-grid and staggered-grid finite-difference methods for frequency-domain acoustic wave modelling: Geophysical Journal International, **157**, 1269–1296.

Ingber, L., and B. Rosen, 1992, Genetic algorithms and very fast simulated reannealing: A comparison: Mathematical and Computer Modelling, **16**, 87–100.

Jackson, C., M. K. Sen, and P. L. Stoffa, 2004, An efficient stochastic bayesian approach to optimal parameter and uncertainty estimation for climate model predictions: Journal of Climate, **17**, 2828–2841.

Jannane, M., W. Beydoun, E. Crase, D. Cao, Z. Koren, E. Landa, M. Mendes, A. Pica, M. Noble, G. Roeth, et al., 1989, Wavelengths of earth structures that can be resolved from seismic reflection data: Geophysics, **54**, 906–910.

Jaysaval, P., D. Shantsev, and S. d. l. K. de Ryhove, 2014, Fast multimodel finite-difference controlled-source electromagnetic simulations based on a schur complement approach: Geophysics.

Jones, M., 2006, Boost application performance using asynchronous i/o.

Kadu, A., T. van Leeuwen, and W. A. Mulder, 2017, Salt reconstruction in full-waveform inversion with a parametric level-set method: IEEE Transactions on Computational Imaging, **3**, 305–315.

Kapoor, S., D. Vigh, H. Li, and D. Derharoutian, 2012, Full waveform inversion for detailed velocity model building: Presented at the 74th EAGE Conference & Exhibition.

Kennedy, J., 2010, Particle swarm optimization, *in* Encyclopedia of Machine Learning: Springer, 760–766.

Kim, Y., C. Shin, H. Calandra, and D.-J. Min, 2013, An algorithm for 3d acoustic time-laplace-fourier-domain hybrid full waveform inversion: Geophysics, **78**, R151–R166.

Komatitsch, D., G. Erlebacher, D. Gddeke, and D. Micha, 2010, High-order finite-element seismic wave propagation modeling with {MPI} on a large {GPU} cluster: Journal of Computational Physics, **229**, 7692 – 7714.

Kopka, H., and P. W. Daly, 2004, Guide to LaTeX: Addison-Wesley.

Krebs, J. R., J. E. Anderson, D. Hinkley, R. Neelamani, S. Lee, A. Baumstein, and M.-D. Lacasse, 2009, Fast full-wavefield seismic inversion using encoded sources: Geophysics, **74**, WCC177–WCC188.

Kronig, R., J. De Boer, and J. Korringa, 1946, On the internal constitution of the earth: Physica, **12**, 245–256.

Kruger, C., 2003, Constrained cubic spline interpolation: Chemical Engineering Applications.

Lailly, P., 1983, The seismic inverse problem as a sequence of before stack migrations: Conference on inverse scattering: theory and application, Society for Industrial and Applied Mathematics, Philadelphia, PA, 206–220.

Lambaré, G., 2008, Stereotomography: Geophysics, **73**, VE25–VE34.

Lamport, L., 1994, LaTeX: A document preparation system: Addison-Wesley.

Lee, D.-T., and B. J. Schachter, 1980, Two algorithms for constructing a delaunay triangulation: International Journal of Computer & Information

Sciences, **9**, 219–242.

Levander, A. R., 1988, Fourth-order finite-difference p-sv seismograms: Geophysics, **53**, 1425–1436.

Lewis, W., B. Starr, and D. Vigh, 2012, A level set approach to salt geometry inversion in full-waveform inversion, *in* SEG Technical Program Expanded Abstracts 2012: Society of Exploration Geophysicists, 1–5.

Lewis, W., and D. Vigh, 2016, *in* 3D salt geometry inversion in full-waveform inversion using a level-set method: 1221–1226.

Li, C., C. Xu, C. Gui, and M. D. Fox, 2010, Distance regularized level set evolution and its application to image segmentation: IEEE transactions on image processing, **19**, 3243–3254.

Liu, F., D. N. Whitmore, D. W. Hanson, R. S. Day, C. C. Mosher, et al., 2004, The impact of reciprocity on prestack source plane wave migration: Presented at the 2004 SEG Annual Meeting, Society of Exploration Geophysicists.

Ma, Y., and D. Hale, 2013, Wave-equation reflection traveltime inversion with dynamic warping and full-waveform inversion: Geophysics, **78**, R223–R233.

Magnoni, F., E. Casarotti, A. Michelini, A. Piersanti, D. Komatitsch, D. Peter, and J. Tromp, 2014, Spectral-element simulations of seismic waves generated by the 2009 laquila earthquake: Bulletin of the Seismological Society of America, **104**, 73–94.

Marfurt, K. J., 1984, Accuracy of finite-difference and finite-element modeling of the scalar and elastic wave equations: Geophysics, **49**, 533–549.

Menke, W., 2012, Geophysical data analysis: discrete inverse theory: Academic press.

Métivier, L., R. Brossier, S. Operto, and J. Virieux, 2015, Acoustic multi-parameter fwi for the reconstruction of p-wave velocity, density and attenuation: preconditioned truncated newton approach, *in* SEG Technical Program Expanded Abstracts 2015: Society of Exploration Geophysicists, 1198–1203.

Métivier, L., R. Brossier, J. Virieux, and S. Operto, 2013, Full waveform inversion and the truncated newton method: SIAM Journal on Scientific Computing, **35**, B401–B437.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, 1953, Equation of state calculations by fast computing machines: The journal of chemical physics, **21**, 1087–1092.

Moody, A., G. Bronevetsky, K. Mohror, and B. R. d. Supinski, 2010, Design, modeling, and evaluation of a scalable multi-level checkpointing system: Proceedings of the 2010 ACM/IEEE international conference for high performance computing, networking, storage and analysis, IEEE Computer Society, 1–11.

Mora, P., 1989, Inversion= migration+ tomography, *in* Parallel Computing 1988: Springer, 78–101.

Morton, S. A., and C. C. Ober, 1998, Fastershot-record depth migrations using phase encoding, *in* SEG Technical Program Expanded Abstracts 1998: Society of Exploration Geophysicists, 1131–1134.

Narang, A., S. Kumar, A. S. Das, M. Perrone, D. Wade, K. Bendiksen, V. Slatten, and T. E. Rabben, 2013, Performance optimizations for tti rtm on gpu based hybrid architectures: Presented at the Biennial International Conference & Exposition.

Nolet, G., 2012, Seismic tomography: with applications in global seismology and exploration geophysics: Springer Science & Business Media, **5**.

Paffenholz, J., J. Stefani, B. McLain, and K. Bishop, 2002, Sigsbee_2a synthetic subsalt dataset-image quality as function of migration algorithm and velocity model error: Presented at the 64th EAGE Conference & Exhibition.

Panagiotakis, C., and A. Argyros, 2016, Parameter-free modelling of 2d shapes with ellipses: Pattern Recognition, **53**, 259–275.

Pica, A., J. Diet, and A. Tarantola, 1990, Nonlinear inversion of seismic reflection data in a laterally invariant medium: Geophysics, **55**, 284–292.

Plessix, R., P. Milcik, H. Rynja, A. Stopin, K. Matson, and S. Abri, 2013, Multiparameter full-waveform inversion: Marine and land examples: The Leading Edge, **32**, 1030–1038.

Plessix, R.-E., 2006, A review of the adjoint-state method for computing the gradient of a functional with geophysical applications: Geophysical Journal International, **167**, 495–503.

Plessix, R.-E., and C. Perkins, 2010, Thematic set: Full waveform inversion of a deep water ocean bottom seismometer dataset: First Break, **28**, 71–78.

Polak, E., and G. Ribiere, 1969, Note sur la convergence de méthodes de directions conjuguées: ESAIM: Mathematical Modelling and Numerical Analysis-

Modélisation Mathématique et Analyse Numérique, **3**, 35–43.

Pratt, R. G., 1990, Frequency-domain elastic wave modeling by finite differences: A tool for crosshole seismic imaging: Geophysics, **55**, 626–632.

Pratt, R. G., and M. Worthington, 1990, Inverse theory applied to multi-source cross-hole tomography. part 1: acoustic wave-equation method: Geophysical prospecting, **38**, 287–310.

Prieux, V., G. Lambaré, S. Operto, and J. Virieux, 2013, Building starting models for full waveform inversion from wide-aperture data by stereotomography: Geophysical Prospecting, **61**, 109–137.

Rao, G., 2001, Sedimentation, stratigraphy, and petroleum potential of krishna-godavari basin, east coast of india: AAPG bulletin, **85**, 1623–1643.

Riyanti, C., Y. Erlangga, R.-E. Plessix, W. Mulder, C. Vuik, and C. Oosterlee, 2006, A new iterative solver for the time-harmonic wave equation: Geophysics, **71**, E57–E63.

Robertsson, J. O., and C. H. Chapman, 2012, An efficient method for calculating finite-difference seismograms after model alterations: Geophysics.

Romero, L. A., D. C. Ghiglia, C. C. Ober, and S. A. Morton, 1999, Phase encoding of shot records in prestack migration: Geophysics, **65**, 426–436.

Saad, Y., 2003, Iterative methods for sparse linear systems: Siam.

Sain, K., and H. Gupta, 2008, Gas hydrates: Indian scenario: Journal of the Geological Society of India, **72**, 299–311.

Sain, K., M. Ojha, N. Satyavani, G. Ramadass, T. Ramprasad, S. Das, and H. Gupta, 2012, Gas-hydrates in krishna-godavari and mahanadi basins: New

data: Journal of the Geological Society of India, **79**, 553–556.

Sajeva, A., M. Aleardi, E. Stucchi, N. Bienati, and A. Mazzotti, 2016, Estimation of acoustic macro models using a genetic full-waveform inversion: Applications to the marmousi model: Geophysics.

Sambridge, M., J. Braun, and H. McQueen, 1995, Geophysical parametrization and interpolation of irregular data using natural neighbours: Geophysical Journal International, **122**, 837–857.

Sambridge, M., and G. Drijkoningen, 1992, Genetic algorithms in seismic waveform inversion: Geophysical Journal International, **109**, 323–342.

Saraswat, P., M. K. Sen, et al., 2010, Simultaneous stochastic inversion of prestack seismic data using hybrid evolutionary algorithm: Presented at the 2010 SEG Annual Meeting, Society of Exploration Geophysicists.

Sava, P., and B. Biondi, 2004a, Wave-equation migration velocity analysis. i. theory: Geophysical Prospecting, **52**, 593–606.

———, 2004b, Wave-equation migration velocity analysis. ii. subsalt imaging examples: Geophysical Prospecting, **52**, 607–623.

Scales, J. A., and M. L. Smith, 1994, Introductory geophysical inverse theory 9drafte.

Schanen, M., O. Marin, H. Zhang, and M. Anitescu, 2016, Asynchronous two-level checkpointing scheme for large-scale adjoints in the spectral-element solver nek5000: Procedia Computer Science, **80**, 1147–1158.

Schuster, G. T., X. Wang, Y. Huang, W. Dai, and C. Boonyasiriwat, 2011, Theory of multisource crosstalk reduction by phase-encoded statics: Geo-

physical Journal International, **184**, 1289–1303.

Sen, M. K., and R. Biswas, 2015, Choice of regularization weight in basis pursuit reflectivity inversion: Journal of Geophysics and Engineering, **12**, 70.

Sen, M. K., and L. N. Frazer, 1985, Kirchhoff-helmholtz reflection seismograms in a laterally inhomogeneous multi-layered elastic medium-ii. computations: Geophysical Journal International, **82**, 415–437.

Sen, M. K., and I. G. Roy, 2003, Computation of differential seismograms and iteration adaptive regularization in prestack waveform inversion: Geophysics, **68**, 2026–2039.

Sen, M. K., and P. L. Stoffa, 1991, Nonlinear one-dimensional seismic waveform inversion using simulated annealing: Geophysics, **56**, 1624–1638.

———, 1992, Rapid sampling of model space using genetic algorithms: examples from seismic waveform inversion: Geophysical Journal International, **108**, 281–292.

———, 2013, Global optimization methods in geophysical inversion: Cambridge University Press.

Sethian, J. A., 1996, A fast marching level set method for monotonically advancing fronts: Proceedings of the National Academy of Sciences, **93**, 1591–1595.

Shah, N., M. Warner, T. Nangoo, A. Umpleby, I. Stekl, J. Morgan, L. Guasch, et al., 2012, Quality assured full-waveform inversion: Ensuring starting model adequacy: SEG Las Vegas 2012 Annual Meeting, http://dx. doi.

org/10, 2012–1228.

Shaw, R., and S. Srivastava, 2007, Particle swarm optimization: A new tool to invert geophysical data: Geophysics, **72**, F75–F83.

Shen, P., and W. W. Symes, 2008, Automatic velocity analysis via shot profile migration: Geophysics, **73**, VE49–VE59.

Shin, C., and D.-J. Min, 2006, Waveform inversion using a logarithmic wavefield: Geophysics, **71**, R31–R42.

Shragge, J., 2014, Reverse time migration from topography: Geophysics.

Sirgue, L., and R. G. Pratt, 2004, Efficient waveform inversion and imaging: A strategy for selecting temporal frequencies: Geophysics, **69**, 231–248.

Snieder, R., M. Xie, A. Pica, and A. Tarantola, 1989, Retrieving both the impedance contrast and background velocity: A global strategy for the seismic reflection problem: Geophysics, **54**, 991–1000.

Song, J.-Y., X.-D. Zheng, Z. Qin, and B.-Y. Su, 2011, Multi-scale seismic full waveform inversion in the frequency-domain with a multi-grid method: Applied Geophysics, **8**, 303–310.

Soubaras, R., 2006, Modulated-shot migration, *in* SEG Technical Program Expanded Abstracts 2006: Society of Exploration Geophysicists, 2430–2434.

Srivastava, S., D. Datta, B. Agarwal, and S. Mehta, 2014, Applications of ant colony optimization in determination of source parameters from total gradient of potential fields: Near Surface Geophysics, **12**, 373–389.

Štekl, I., and R. G. Pratt, 1998, Accurate viscoelastic modeling by frequency-domain finite differences using rotated operators: Geophysics, **63**, 1779–

1794.

Stoffa, P. L., M. K. Sen, R. K. Seifoullaev, R. C. Pestana, and J. T. Fokkema, 2006, Plane-wave depth migration: Geophysics, **71**, S261–S272.

Stolt, R., 1978, Migration by fourier transform: Geophysics, **43**, 23–48.

Storn, R., and K. Price, 1997, Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces: Journal of global optimization, **11**, 341–359.

Stumm, P., and A. Walther, 2010, New algorithms for optimal online checkpointing: SIAM Journal on Scientific Computing, **32**, 836–854.

Subrahmanyam, V., A. Subrahmanyam, G. Murty, and K. Murthy, 2008, Morphology and tectonics of mahanadi basin, northeastern continental margin of india from geophysical studies: Marine Geology, **253**, 63–72.

Sun, P., S. Zhang, and F. Liu, 2002, Prestack migration of areal shot records with phase encoding, *in* SEG Technical Program Expanded Abstracts 2002: Society of Exploration Geophysicists, 1172–1175.

Sun, X., and S. Suh, 2011, Maximizing throughput for high performance tti-rtm: From cpu-rtm to gpu-rtm, *in* SEG Technical Program Expanded Abstracts 2011: Society of Exploration Geophysicists, 3179–3183.

Symes, W., and J. J. Carazzone, 1991, Velocity inversion by differential semblance optimization: Geophysics, **56**, 654–663.

Symes, W. W., 2007, Reverse time migration with optimal checkpointing: Geophysics, **72**, SM213–SM221.

———, 2008, Migration velocity analysis and waveform inversion: Geophysical prospecting, **56**, 765–790.

Tao, Y., and M. K. Sen, 2012, Frequency-domain full waveform inversion with plane-wave data: Geophysics, **78**, R13–R23.

Tarantola, A., 1984, Inversion of seismic reflection data in the acoustic approximation: Geophysics, **49**, 1259–1266.

———, 2005, Inverse problem theory and methods for model parameter estimation: siam.

Varela, C. L., P. L. Stoffa, and M. K. Sen, 1998, Background velocity estimation using non-linear optimization for reflection tomography and migration misfit: Geophysical Prospecting, **46**, 51–78.

Vigh, D., and E. W. Starr, 2008, 3d prestack plane-wave, full-waveform inversion: Geophysics, **73**, VE135–VE144.

Virieux, J., 1986, P-sv wave propagation in heterogeneous media: Velocity-stress finite-difference method: Geophysics, **51**, 889–901.

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

Wang, H., S. C. Singh, H. Jian, and H. Calandra, 2012, Integrated inversion of subsurface velocity structures using wave equation tomography and full waveform inversion, *in* SEG Technical Program Expanded Abstracts 2012: Society of Exploration Geophysicists, 1–5.

Wang, Q., P. Moin, and G. Iaccarino, 2009, Minimal repetition dynamic checkpointing algorithm for unsteady adjoint calculation: SIAM Journal on Sci-

entific Computing, **31**, 2549–2567.

Warner, M., and L. Guasch, 2014, Adaptive waveform inversion-fwi without cycle skipping-theory: Presented at the 76th EAGE Conference and Exhibition 2014.

Wright, S. J., and J. Nocedal, 1999, Numerical optimization: Springer New York, **2**.

Wu, Z., and T. Alkhalifah, 2014, Full waveform inversion based on the optimized gradient and its spectral implementation: Presented at the 76th EAGE Conference and Exhibition 2014.

Xia, G., M. K. Sen, and P. L. Stoffa, 1998a, 1-d elastic waveform inversion: A divide-and-conquer approach: Geophysics, **63**, 1670–1684.

———, 1998b, 1-d elastic waveform inversion: A divide-and-conquer approach: Geophysics, **63**, 1670–1684.

Zelt, C., 1997, Rayinvr: Documentation and related programs, version 1.4. 2.; manual: Rice University, Texas, USA.

Zelt, C., and R. Smith, 1992, Seismic traveltime inversion for 2-d crustal velocity structure: Geophysical journal international, **108**, 16–34.

Zhang, Y., and D. Wang, 2009, Wave-phase information based full waveform inversion: Presented at the 71st EAGE Conference & Exhibition.

Zhang, Z.-d., and T. Alkhalifah, 2017, Full waveform inversion using oriented time-domain imaging method for vertical transverse isotropic media: Geophysical Prospecting.

Zhou, H., G. Zhang, and R. Bloor, 2006, An anisotropic acoustic wave equation

for modeling and migration in 2d tti media, *in* SEG Technical Program Expanded Abstracts 2006: Society of Exploration Geophysicists, 194–198.

Zhu, C., R. H. Byrd, P. Lu, and J. Nocedal, 1997, Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization: ACM Transactions on Mathematical Software (TOMS), **23**, 550–560.

Vita

Debanjan Datta was born in the city of Kolkata, India. He received an Integrated Masters of Science and Technology degree from the Indian School of Mines (Now Indian Institue of Technology, ISM Dhanbad) , Dhanbad, India. Right after he graduated from Indian School of Mines, Debanjan applied to the University of Texas at Austin in 2013 for enrollment into the Ph.D program at the Jackson School of Geosciences. He was accepted and since then has worked as an Graduate Research Assistant at the department. He is a candidate for the degree of Doctor of Philosophy at the The University of Texas at Austin.

Permanent address: debanjand@utexas.edu

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.