THE SEPARATION AND IMAGING OF CONTINUOUSLY RECORDED SEISMIC DATA

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

Christopher Leader
December 2015

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

(Biondo L. Biondi) Principal Adviser

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

(Robert Clapp)

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

(Gerald Mavko)

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

(Jon F. Claerbout)

Approved for the University Committee on Graduate Studies

Abstract

Imaging targets for exploration seismology surveys are increasingly associated with subsurface conditions that pose serious challenges for both seismic model building and imaging. In particular, these zones of interest are often sub-salt, and can feature dipping, anisotropic overburdens. In order to construct representative Earth models, and hence an accurate image, reflection datasets with large offsets, high levels of reflection point redundancy, and rich azimuthal coverage are sought. This creates a requirement for denser and more efficient field acquisition techniques, which can be achieved through blended data acquisition. Here, multiple source points are acquired simultaneously and receivers continuously record the overlapping wavefields. Separating these simultaneously acquired seismic data into their conventionally acquired equivalent is the link between more efficient data acquisition and conventional imaging and model building techniques.

The focus of this thesis is the development of a robust methodology for taking these continuously recorded, blended datasets and accurately separating each shot into an individual record, free from blending contamination. Successful existing techniques unanimously feature strict requirements on the randomness of the source distribution during acquisition, as a consequence, a technique which can separate data with repeatable delay patterns is demonstrated.

By using extended seismic migration and inverse forward modeling, separation is demonstrated which is less sensitive to shooting pattern. The image space has several features amenable to the separation problem; it is of lower dimensionality, it has a higher signal-to-noise ratio and methods such as preconditioning and regularization

can be employed. Images with a reduced quantity of interference-related noise are created, then a forward modeling process provides a separated output dataset. While forward modeling alone provides a high level of separation, amplitudes subleties of these input data can be lost. By posing the data recovery process as an inverse problem, representative amplitude character is recovered.

Posing the problem in the extended image space relaxes requirements on the accuracy of the migration velocity model. This extended space preserves all kinematic and amplitude information present in the input data, and there is no loss of information during imaging. This methodology is demonstrated on three synthetic datasets, of varying complexity, and a 3D Ocean Bottom Node dataset. Qualitative and quantitative comparisons for the synthetic tests show the separation is accurate to the order of 5% after ten iterations. Qualitative analysis of the OBN separation shows that comparable images are achieved after separation, compared to an unblended input dataset.

Preface

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

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

[1] http://sepwww.stanford.edu/private/docs/sep146
[2] http://sepwww.stanford.edu/public/docs/sepdatalib/toc_html

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

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

Our testing is currently limited to LINUX 2.6 (using the Intel Fortran90 compiler) and the SEPlib-6.4.6 distribution, but the code should be portable to other architectures. Reader's suggestions are welcome. For more information on reproducing SEP's electronic documents, please visit *http://sepwww.stanford.edu/research/redoc/*.

Acknowledgments

Nothing can really prepare you for what life as a PhD student will be like. Several aspects were easier than anticipated, and several were much more difficult. Becoming accustomed to the fact that, more often than not, the only impedance to my progress was myself, took some time. Whether it was asking other students for help, talking to advisors, working more hours, reading more papers, conducting more tests, drinking more coffee, taking a break (whether deserved or undeserved) and so on, developing a working strategy for independent research is an iterative process in itself. Reflecting on the six year process, as a whole, is a very rewarding process. During this reflection, it is more clear than ever that I owe a large number of people a great deal of thanks for making this experience possible.

Naturally, the first person to thank is Professor Biondo Biondi, for accepting me to SEP and guiding me through the entire process. I applied late, on a previous advisor's recommendation, but fortunately Biondo took the risk. Despite his myriad of students, numerous Stanford-wide responsibilities and busy family life he would always find time to meet when it was necessary, and in these last few months has been instrumental in expediting data and scheduling issues. Secondly, a huge amount of thanks goes to Bob Clapp. Without his help, I'd still be trying to get my inversion code to pass the dot-product test. His expertise in computational geophysics has been an indescribable asset, and without his help and knowledge then I would not have the next job that I'm about to begin. Huge thanks also to the founder of SEP, Jon Claerbout, for ideas, discussions, lunches, breakfasts, pool parties, and for the legacy he has created at Stanford.

I would like to thank the sponsors of SEP over the last six years for providing the financial resources, and a number of additional tools, that have assisted in my work. In particular I have to thank Scott Morton, of Hess Corporation, who quite literally saved me this year after a number of field data sources fell through. He put significant effort into helping me gain access to data and computational resources, and graduating this calendar year may have not been possible without his help.

Particular thanks also to NVIDIA, for providing many of the GPUs I ran my results on, and to Paulius Micikevicius, formerly of NVIDIA, for sharing codes and thoughts with me over the years. Without some of his suggestions, getting multi-GPU wave propagation running without errors would have been a much more arduous process. Thanks also to Colin Perkins and Laurent Sirgue, of Shell and Total respectively, who oversaw my summer internships.

Senior SEP students, who have all now graduated, all played a role in my development as a geophysicist - so thanks goes to Claudio Cardoso, Kittinat Taweesintananon, Yaxun Tang, Gboyega Ayeni, Qiang Fu, Sjoerd de Ridder, Nader Moussa, Adam Halpert, Yunyue Li, Mandy Wong, Xukai Shen and Yang Zhang. As did junior students, who pushed my knowledge of seismology and SEPlib, and kept trips to sponsor meetings and SEG enjoyable - so, thanks goes to Ohad Barak, Ali Almomin, Yi Shen, Musa Maharramov, Taylor Dahlke, Huy Le, Eileen Martin, Guillaume Barnier, Gustavo Alves, Ettore Biondi, Alejandro Cabrales, Yinbin Ma and Jason Chang (the eternal first year.)

In particular, thanks to Xukai, Nader, Eileen and Ali, who have been my office mates (in some combination) on the 4th floor. You all helped with work problems, sanity problems, and many others. Also, thanks to Jason for typing when me and Andreas Mavromatis played Sporcle, and to Emily Fay and Taylor, for letting me, Andreas and Jason use your office space for Sporcle requirements.

Housemates over the years all assisted to making this Californian existence rewarding - Jumie Yuventi, Josie Nevitt, Jeremy Brown, Ben Flatau, Zachary Brown, Mary Davis, Luke Semo, Rodney Ruiz, Andrew Siak, Katie MacKenzie, Emily Hsiung

and Pam Wu (wow, that's a lot...), thanks for living with me at various points.

Thanks to the close friends I made too - for trivia nights, parties, band practices and more - Adam Allan, Denys Grombacher, Kevin Seats and many more.

Finally, thanks to my family, for the constant support they have given me throughout this whole process. It would have been much more difficult without you all.

Contents

Abstract **v**

Preface **vii**

Acknowledgments **ix**

1 Introduction **1**

2 Wave-equation imaging and linearized inversion **13**

3 Phase-encoded inversion and randomized sampling **39**

4 Simultaneous shot separation **57**

5 Field data example **113**

6 High performance computing solutions **135**

Bibliography **159**

List of Tables

2.1 Wave equation coefficients for various geophysical methods 14

List of Figures

1.1 Rough plots of four acquisition techniques - Narrow Azimuth Towed Streamer (NATS), Multiple Azimuth Acquisition (MAZ), Wide Azimuth Towed Stream (WATS) and Coil shooting. [**NR**] 3

1.2 A dataset acquired with random delays between overlapping sources. The left panel is an example shot gather, the right panel, an example receiver gather. [**CR**] 5

1.3 A dataset acquired with constant delays between overlapping sources. [**CR**] 6

2.1 Representation of a centred, symmetric, 8th order finite-difference stencil. [**NR**] 16

2.2 Section of the SEAM velocity model, used for modeling and imaging. [**ER**] 23

2.3 The scattering potential to be imaged, calculated by subtracting a lightly smooth version of the velocity model from itself. [**ER**] 24

2.4 An example of a 3D shot simulated using linearized forward modeling over the proposed velocity model. [**CR**] 25

2.5 The unfiltered RTM image produced from the synthetic data shown above. For this image the source domain was well sampled with 800 inline shots and 30 crossline shots. [**CR**] 26

2.6 The recovered scattering model after a single iteration of conventional linearized inversion. [**CR**] 28

2.7 The recovered scattering model after ten iterations of conventional linearized inversion, clipped identically to Figure 2.6. [**CR**] 29

2.8 Normalized data misfit as a function of iteration number. Measured from the SEAM model above. [**NR**] 31

2.9 The first gradient of linearized inversion without any preconditioning. [**CR**] 33

2.10 The first gradient of linearized inversion preconditioned with a vertical derivative. [**CR**] 34

2.11 The first gradient of linearized inversion preconditioned with a Laplacian. [**CR**] 35

2.12 How each preconditioned system performs as a function of iteration, measured by L2 data misfit. [**NR**] 36

3.1 The result after one iteration of phase encoded inversion, (a), and after 50 iterations, (b). Here 120 shots were combined into one supershot. [**CR**] 44

3.2 The result after one iteration of LSRTM, (a), and after 50 iterations of phase encoded inversion, (b). [**CR**] 46

3.3 The result after ten iterations of LSRTM, (a), and after 50 iterations of phase encoded inversion, (b). [**CR**] 47

3.4 How the data fit improves as a function of iteration number, for linearized inversion and for phase encoded inversion. [**NR**] 48

3.5 The same plot as Figure 3.4, but as a function of cost. The cost is normalized to that of one iteration of LSRTM, with results for LSRTM linearly interpolated for the sake of representation. [**NR**] 49

3.6 Images obtained after one iteration using 75%, 50% and 25% of the data, respectively. [**CR**] 51

3.7 Images obtained after ten iterations using 75%, 50% and 25% of the data, respectively. [**CR**] 52

3.8 Images obtained after 50 iterations using 75%, 50% and 25% of the data, respectively. [**CR**] 53

3.9 Plot of how the data misfit varies as a function of iteration number. [**NR**] 54

3.10 The same plot as in Figure 3.4, but as a function of cost. [**NR**] . . . 55

4.1 Five randomly delayed shots, while the record is noisy it is possible to distinguish which shot caused which event. [**CR**] 62

4.2 Two shots, simulated over the same model as Figure 4.1. At late times, it becomes impossible to ascertain which shots given events originated from. [**CR**] 63

4.3 A simple dataset, from a two layer model. The left side panel is a shot gather example, the right side panel is a receiver gather. [**CR**] 63

4.4 The same dataset as in Figure 4.3, after random blending (top) and linear blending (bottom). [**CR**] 64

4.5 The unblended simple dataset, as seen in Figure 4.3, migrated into the extended domain with the correct velocity. [**CR**] 69

4.6 The unblended simple dataset, as seen in Figure 4.3, migrated into the extended domain with using a constant velocity model. [**CR**] 70

4.7 Accurate and smoothed representations of the Marmousi velocity model. [**ER**] 72

4.8 Data acquired over the Marmousi model. [**CR**] 73

4.9 Data acquired over the Marmousi model using a random delay function. [**CR**] 74

4.10 The extended image from migrating Figure 4.9 (data acquired using a random blending function) [**CR**] 75

4.11 Data acquired over the Marmousi model using a linear delay function. [**CR**] 76

4.12 The extended image from migrating Figure 4.11. [**CR**] 77

4.13 Data acquired over the Marmousi model using a pseudo-linear delay function. [**CR**] 79

4.14 The extended image from migrating Figure 4.13 (data acquired using pseudo-linear delays). [**CR**] 80

4.15 Linearly blended data migrated into the subsurface offset domain using the correct velocity model (top) and a model 10% too slow (bottom.) [**CR**] 82

4.16 The same images as Figure 4.15 but with the third axis transformed into the subsurface angle domain, rather than subsurface offset. [**CR**] 83

4.17 The angle domain image with the correct velocity model after a single parabolic transform (top) and the slow velocity image after a single parabolic radon transform. [**CR**] 84

4.18 The angle domain image with the correct velocity model after a single parabolic transform (top) and after ten linear iterations of the transform (bottom.) Note the focusing at zero curvature (corresponding to a flat angle gather.) [**CR**] 85

4.19 The ouput dataset from one pass of forward modeling, using the correct velocity model. [**CR**] 87

4.20 Inverse Born modeling using the correct velocity model, after ten iterations. [**CR**] 90

4.21 Adjoint Born modeling using an incorrect velocity model, after ten iterations. [**CR**] 91

4.22 Adjoint Born modeling using an incorrect velocity model, after ten iterations. [**CR**] 92

4.23 Convergence of inverse Born modeling as a function of iteration number. The normalised residual is measured in the image space. [**NR**] . 93

4.24 The output dataset after applying one pass of Born modeling to Figure 4.10, which was the image created from a randomly delayed dataset. [**CR**] 95

4.25 The output dataset after applying one pass of Born modeling to Figure 4.12, which was the image created from a linearly delayed dataset. [**CR**] 96

4.26 The output dataset after applying one pass of Born modeling to Figure 4.14, which was the image created from a pseudo-linearly delayed dataset. [**CR**] 97

4.27 The extended image from migrating Figure 4.13 (data with constant delays) using a rough velocity model. [**CR**] 98

4.28 The velocity model windowed from the SEAM model, used for a more realistic separation test. [**ER**] 101

4.29 A conventional dataset acquired using the section of the SEAM model. [**CR**] 102

4.30 A heavily smoothed version of Figure 4.28, used for inaccurate model separation. A rectangle filter of 30 model points x 30 model points was used. [**ER**] 103

4.31 A pseudo-linearly blended dataset acquired using the section of the SEAM model. [**CR**] 104

4.32 The extended image created by migrating the blended dataset with the velocity model shown in Figure 4.30. [**CR**] 105

4.33 The output separated data after 10 iterations of inverse deimgration using the correct velocity model. [**CR**] 106

4.34 The output separated data after 10 iterations of inverse deimgration using an inaccurate velocity model. [**CR**] 107

4.35 Images created from conventionally acquired data, blended data separated using the correct velocity, and blended data separated using an inaccurate velocity model respectively. 108

4.36 Thirty shots migrated into the extended domain before deblending, using the inaccuarte separation velocity. [**CR**] 109

4.37 Thirty shots migrated into the extended domain after deblending, using the inaccurate separation velocity. [**CR**] 110

4.38 How the separation algorithm performs, for SEAM, as a function of iteration number. This normalised residual is the L2 norm of the difference between the input, blended data, and the output, separated data after reblending. [**NR**] 111

5.1 A diagram of upgoing and downgoing rays for OBN acquisition. [**NR**] 115

5.2 A diagram of how only downgoing rays can be simulated, and the principle of reciprocity. [**NR**] 115

5.3 A rotated section of the RTM image, computed from the 103 OBNs. [**NR**] 117

5.4 A 3D image at the same coordinates as Figure 5.3, after five iterations of LSRTM. [**NR**] 118

5.5 The same result as Figure 5.3, with AGC applied. [**NR**] 119

5.6 The same result as Figure 5.4, with AGC applied. [**NR**] 120

5.7 An example receiver gather after five OBNs are blended together. [**NR**] 122

5.8 An example input receiver gather for the separation engine, which is a section of Figure 5.7 after Γ has been applied. [**NR**] 123

5.9 Isotropic, extended migration of these blended data using the correct velocity model. [**NR**] 124

5.10 Isotropic, extended migration of these blended data using an incorrect velocity model. [**NR**] 125

5.11 From left to right: an (unblended) receiver gather, the reconstructed gather using extended image space separation and an incorrect velocity model and the reconstructed gather using zero-offset, isotropic image space separation. [**NR**] 126

5.12 A raw, unblended receiver gather. [**NR**] 126

5.13 A receiver gather after extended unblending. [**NR**] 127

5.14 A receiver gather after unextended, isotropic unblending. [**NR**] . . . 128

5.15 TTI migration of the input data before blending, for a baseline comparison. [**NR**] 129

5.16 TTI migration of the reconstructed data after 10 iterations of zero-subsurface-offset, isotropic deblending. [**NR**] 130

5.17 TTI migration of the reconcstructed data after 10 iterations of extended, istropic deblending with a rough velocity model. [**NR**] 131

5.18 An inline comparison of the images shown in Figure 5.15, Figure 5.16 and Figure 5.17. [**NR**] 132

6.1 Diagram of the memory layout of a typical multi-core CPU node. As distance from the Floating Point Unit (FPU) increases, so does both the size and latency of the memory unit. [**NR**] 137

6.2 Compute speed (million models pts calculated per second) against implementation. [**NR**] 138

6.3 Compute speed (million models pts calculated per second) against implementation. [**NR**] 140

6.4 Schematic for a given GPU architecture, demonstrating how the grid is broken into two-dimensional thread blocks, which are broken up into individual threads. [**NR**] 142

6.5 A visual representation for the different memory levels within a GPU. Each block has a unique shared memory and each thread has it's individual local memory and set of registers. Global, constant and texture memories are shared between all thread blocks. [**NR**] 142

6.6 Diagram of warp alignment with memory requests. The above two cases feature 100% bus utilization, the bottom case has 80% due to misaligned memory accesses. [**NR**] 145

6.7 Compute speed (million models pts calculated per second) against implementation. [**NR**] 146

6.8 Visual representations of how I/O balances relative to other operations for four GPU based RTM scenarios: disk based wavefield access, asynchronous disk based wavefield access, CPU DRAM based wavefield access and GPU based random-boundary wavefield re-computation. [**NR**] 149

6.9 Breaking the computational domain across multiple GPUs, dark grey indicates the overlapped area between neighboring GPUs. [**NR**] . . . 151

6.10 Staggered representation for halo region transfer. Initially all GPUs send to the 'right' and receive from the 'left,' then the order is reversed. [**NR**] 152

6.11 How the individual linearized inversion operations scale with the number of GPUs used, relative to normalized computational speeds. [**NR**] 153

Chapter 1

Introduction

Whilst advanced imaging algorithms have proved to be a crucial tool for many imaging problems, the hindrance for imaging deep and difficult targets is often associated with data coverage. Effective seismic imaging and model building in 3D hinges on several key factors, such as the offset range (determining the rough maximum depth of imaging), the azimuthal coverage, and the density of sampling (for both sources and receviers).

Many acquisition solutions exist that aim to improve offset coverage, azimuthal coverage, and sampling density, however a significant increase in acquisition cost is inherent. These solutions may feature multiple acquisition vessels, multiple source vessels, or varied shooting patterns, but the factor that increases survey cost the most dramatically is time. A typical survey crew costs roughly $500,000 per day, with 3D surveys often lasting 3-6 months. The costs incurred while shooting these data are enormous, and so methods of increasing suvery efficiency are of huge interest. Additionally, these datasets are extremely large, and expand with increase in offset, azimuth and sampling density. Consequently, the imaging approach used must often be adapted to accomodate these huge data quantities.

A typical Narrow Azimuth Towed Streamer (NATS) acquisition vessel (Figure 1.1, top left) features a typical maximum offset of 10 km in the inline direction, and 1

km in the crossline direction. Typical receiver sampling intervals are 25 m and 50 m in the inline and crossline directions respectively, the source sampling is typically comparable. The rectangular patch indicates the subsurface illumination achieved with this configuration. The sampling density NATS is reasonable, however the offset range limits the depth of imaging to 10 km (for flat geology) and both the azimuthal richness and the crossline offset are very low. Since seismic model building techniques rely on redundant reflection point information in the subsurface, in terms of offset and azimuth, then accurately building these models in the crossline direction becomes difficult.

One method is to reacquire the survey along a different azimuth. The azimuthal range of these data increases linearly with azimuths acquired, but so does survey time, and hence cost. Another method is Wide Azimuth Towed Stream (WATS), also shown in Figure 1.1. As many as four extra source boats are used, greatly increasing subsurface illumination, by virtue of both offset and azimuth (again, indicated by the rectangle.)

A WATS type approach has been shown to produce excellent images, Verwest and Lin (2007), but a major bottleneck is the increase in survey time. After each shot point has been fired a sufficient waiting period is elapsed before the next shot point, so that energy has sufficiently dissipated. Typical imaging algorithms assume a single shot point per record, so high-amplitude and coherent overlapping arrivals between sources will be very damaging imaging results. Often 15 seconds are elapsed before the next point is acquired, so for a survey comprising of four source boats, a full minute must elapse before a given source boat acquires it's next location.

If this waiting period restriction was relaxed, many more source points could be acquired in a given time interval (Beasley et al., 1998a). This concept has the potential to drastically reduce survey time, and the financial savings can be drastic. Acquiring data in this manner is referred to as simultaneous source data, continuously recorded data, and blended data, amongst others. The recorded data will feature significant, high amplitude, overlapping arrivals between shot records, and this must be taken into account during processing, model building and imaging. Naturally, this will

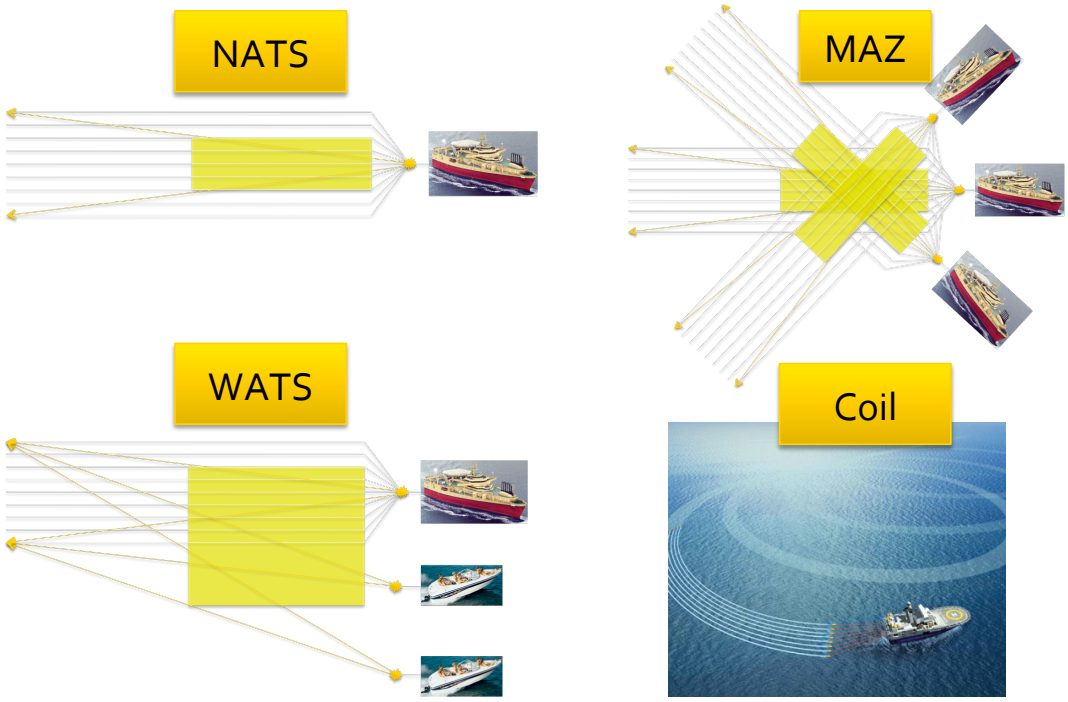

Figure 1.1: Rough plots of four acquisition techniques - Narrow Azimuth Towed Streamer (NATS), Multiple Azimuth Acquisition (MAZ), Wide Azimuth Towed Stream (WATS) and Coil shooting. [**NR**] chap1/. acq

make the data processing more expensive, but this increase in processing cost will be drastically less than the money saved through quicker acquisition.

BLENDED DATA

There are numerous opportunites, and challenges, provided by blended acquisition. A number of studies have looked at extracting subsurface properties directly from the overlapping records, such as Tang and Biondi (2009), however these approaches implicitly require an accurate velocity model. If blended surveying is to become an option for exploration type surveying, then processing techniques which do not rely on prior subusurface knowledge are essential.

Developing a methodology that can effectively separate these overlapping shot gathers into their conventionally acquired equivalent records would provide the link between efficient data acquisition, and contemporary model-building and imaging work flows. Thus, the focus of this thesis will be on data separation, rather than direct imaging of these data. There are a number of existing separation techniques, and many of them work extrememly well. In particular, the work by Abma et al. (2010) is now used widely by BP. Doulgeris et al. (2010), Ayeni and Biondi (2009) and Moore et al. (2008) all have demonstrated useful separation procedures, by exploiting the randomness of source points acquired.

Figure 4.9 shows a 2D dataset where randomly delayed overlap between sources has been permitted. By studying the receiver gather panel, it is clear that this overlapping energy is randomly scrambled in this domain. A selection of random noise attenuation methodologies can now be employed, and this is how the mentioned authors achieved their separation. However, Figure 4.11 shows these same data, but with constant delays between sources. There is now no coherency difference between the source and receiver domains; attempting to apply one of these mentioned techniques would now entirely fail.

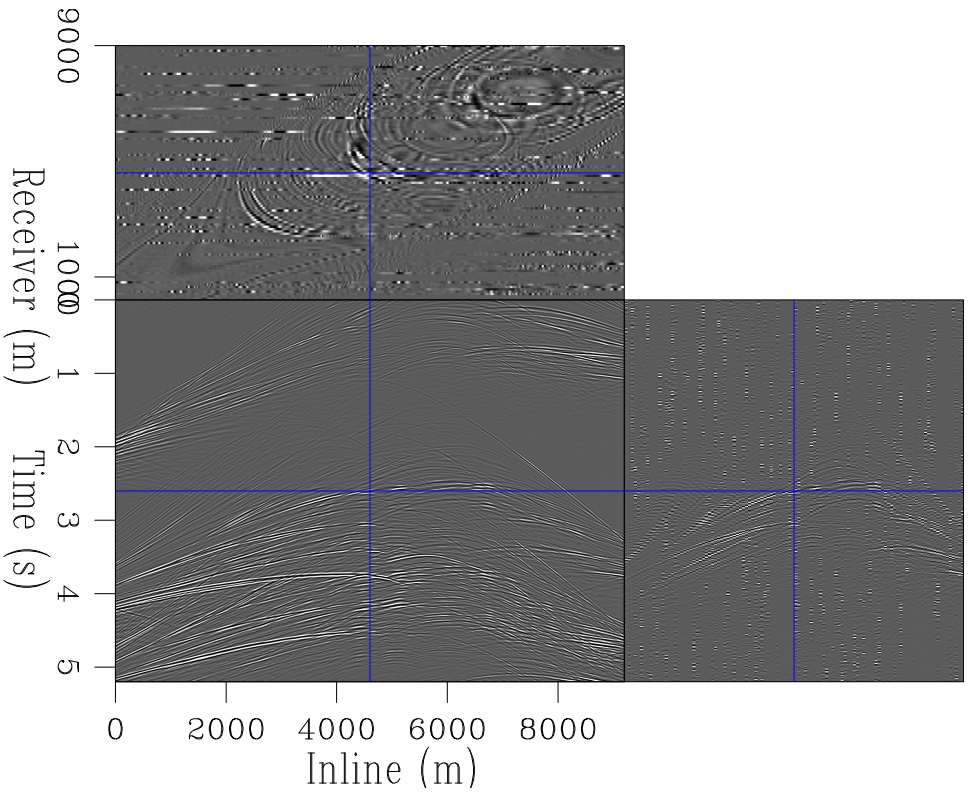

Figure 1.2: A dataset acquired with random delays between overlapping sources. The left panel is an example shot gather, the right panel, an example receiver gather. [**CR**] chap1/. marmrndblnd

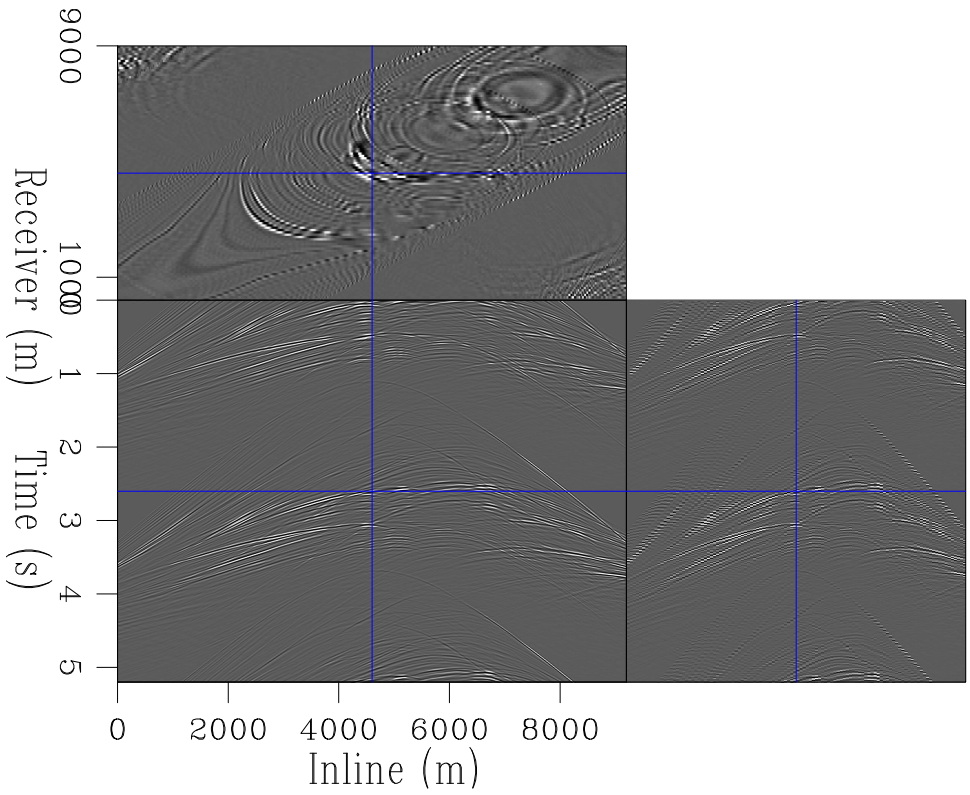

Figure 1.3: A dataset acquired with constant delays between overlapping sources. [**CR**] chap1/. marmlinblnd

Image space separation

These successful data-space approaches use either Fourier transforms, Radon transforms or wavelet transforms to move to a domain where coherency attributes can be isolated. By a series of domain transforms, or by an inverse process, random noise is iteratively removed and clean receiver gathers formed. Subsequently, by the principle of reciprocity, separated shot gathers have been created.

It is also possible to transform to the extended image domain, rather than the Fourier, Radon or wavelet domain. The image domain is often usefully employed, since it is of much lower dimensionality, and has a high signal-to-noise ratio, when compared to the data-space. This image space transform, known as migration, requires a subsruface velocity model. Instead, by transforming to the extended image domain, the requirement of an accurate velocity model is relinquished. Kinematic and amplitude information contained within these data are all preserved. It is possible to manipulate these extended images, according to a set of criteria, and to then transform back to the data-space. Weibull and Arntsen (2014) used a similar method for surface-related multiple removal.

This thesis will use the imaging transform as a type of coherency filter, and separate overlapping data using the image domain. This approach is effective for both randomly delayed shots and constantly delayed shots, since even for consistent delays the artifacts are not coherent between separate images. Chapter 4 will formally introduce how an inverse forward modeling procedure can be formulated, for accurate data separation under a variety of blending methodologies.

Artificial blending

As alluded to above, an increase in acquisition complexity leads to an increase in the quantity of data acquired. For data acquired in an unblended fashion, subsequent, artificial blending can be a useful endeavour. This is known as phase encoding.

For seismic imaging and inversion, shots are migrated and modeled invidually,

and then combined. It is possible to delay and sum shot gathers together, and then image and model them concurrently (Romero et al., 2000). Such a methodology induces crosstalk artifacts, but these can be reduced according to how shots are scaled and summed together. Furthermore, with stacking these artifacts will be greatly diminished, while real events will actively sum. The difference between phase encoded and field blended data is that the processor has direct control over how the shots are combined.

Chapter 3 will analyse two methods of data space reduction - phase encoding, and randomized sampling, whereby rolling subsets of the data are selected during separate iterations of an inverse process. This is in contrast to the different set of tools needed to process field-blended data, and provides a meaningful discussion and comparison between field blending and artificial blending.

HPC METHODS

The extended imaging, demigration, and inversion approaches applied throughout this thesis are extrememly computationally intensive. Simulating a simplified form of the wave-equation is the engine of all these methodologies, and this simulation requires millions of finite-difference stencil convolutions to apply the spatial derivative.

To account for this demand, a significant portion of the research and work necessary for this thesis was in accelerating wave-equation imaging and modeling. The main focus of this was in the use of Graphical Processing Units (GPUs) to accelerate this stencil convolution. GPUs are an example of Single Instruction Multiple Data (SIMD) computing, where a single set of instructions (a kernel) are applied to a large array, or set of arrays, very efficiently. A GPU, at base level, can be considered as several hundred (to several thousand) mini CPUs, each acting independently, and each with an individual, and small, low latency memory level. Acoustic wave propagation is a close to ideal candidate for GPUs, since the bulk of the computation can be made to fit this SIMD framework.

GPU computing poses challenges, however. Each individual card features a small

global memory, so for meaningful computation an array of GPUs must be used in parallel, and communication between these units is paramount. Chapter 6 will discuss GPU computing in detail, and elaborate on many of these details.

It should be noted that the vast majority of results contained in this thesis use a GPU based code library for wave propagation, thus most results are labelled Conditionally Reproducible (CR). This is because an available GPU card with compute capability of CUDA 4.0+ is required for the bulk of the computation, despite run times usually being brief. The codes have been tested and validated on M2090, K10 and K40 cards.

THESIS OVERVIEW

Wave-equation imaging and linearized inversion: Chapter 2 introduces the core concepts of wave-equation imaging, and it's extension to linearized inversion. These techiques are used throughout the entire thesis, and so a formal introduction is necessary. With particular reference to Reverse Time Migration, results from a complex synthetic model are shown for adjoint imaging and for data-space inversion. Quantitive measures of these effectiveness of these methods are introduced and compared, and methods for accelerating the process discussed. The concept of preconditioning is discussed, and the use of a few simple operators are employed, with the goal of increasing inversion efficiency, as a function of cost. This will be contrasted later in the thesis, where regularization methods and HPC methods are used to improve and accelerate linearized inversion.

Phase-encoding and randomized sampling: In Chapter 3 two methods of data-space reduction are introduced - phase-encoding and randomized sampling. The concepts of both static and dynamic phase-encoding are discussed, the algorithms contrasted, and a set of inversion results shown. Residual reductions are plotted as functions of both iteration number and cost, and the improvement of convergence discussed. Phase-encoded is then contrasted with randomized sampling, where a rolling subset of the data is used between iterations. Results

are plotted as functions of iteration and cost, again. Results of augmenting these two methodologies are then mentioned.

Simultanous shot separation: Chapter 4 contains much of the core work of the thesis. Existing separation techniques, and their drawbacks, are discussed at length, and then the concept of image space separation is introduced. The domain coherency differences of a variety of blending techniques shown clearly, and the manifestation of these different delays in the extended image space, for a simple and complicated synthetic model, are shown. Isolating events according to the curvature by using a parabolic Radon inversion in the angle domain is demonstrated, and the results show strong event separation. The original concept of extended demigration for shot separation is formally presented, as is it's extension to an inverse forward modeling process. The concept of incorporating a blending operator into an image-space deblending technique is demonstrated. Separation results using this innovative technology are presented, for both accurate and inaccurate velocity models, over a range of three synthetic models of increasing complexity. Quantitiative measures of the effectiveness of this concept are used, and the results show good convergence.

Field data example: In Chapter 5, several of the concepts discussed in Chapter 2 and Chapter 4 are applied to a field dataset. This dataset featured Ocean Bottom Node receivers, and a set of anisotropic models. This required a non-negligible extension of these procedures, and the results of linearized inversion show large improvements. These data are then blended and deblended, using an acoustic deblending engine, with accurate and inaccurate velocity models. The images constucted from these separated data are directly comparable, however these separate receiver gathers show a number of artifacts. Possible sources of these are then discussed, and improvements for the procedure suggested.

High performance computing solutions: Finally, Chapter 6 presents all the research and work that went into building the wave-equation engine that created the results for all previous chapters. Both CPU and GPU based linearized

inversion solutions are discussed and contrasted, and the requirement of different source wavefield domain boundaries mentioned. The extension to multiple GPU propagation is then demonstrated in detail, for propagation, RTM and Born modeling. How the necessary extension to anisotropic propagation was done (to construct the results of Chapter 5) is then mentioned, with reference to how GPU memory and storage structures were made use of.

Chapter 2

Wave-equation imaging and linearized inversion

Developing a methodology to correctly reposition recorded seismic reflection events is one of the most crucial aspects of exploration seismology. As a result, imaging algorithms have been an intensely researched topic over the last several decades. The underlying physics of seismic scattering, while complicated, are well understood; it is the computational feasibility of accurate wave simulation that remains a major bottleneck in image construction

Increases in computing power over the last two decades have brought forth the ability to, within certain assumptions, calculate direct solutions to the wave-equation. Accurate simulation of the two-way wave-equation is vital to capture many wave-field complexities, such as multiply scattered events, prismatic events, attenuation, overturned waves, elastic arrivals, mode conversions, anisotropy and steeply dipping arrivals (Alford et al. (1974); Mulder and Plessix (2004); Marfurt (1984)). Earlier efforts to numerically solve the wave-equation, so called one-way extrapolators, could not reliably track any of these aforementioned complexities (Claerbout, 1971).

However, simplifications of the physics are still necessary, and many assumptions remain inherent. As computational power increases, so does the ability to capture

Method	**u**	h	g
Seismic Reflection	Pressure/displacement	$1/c^2$	$2\alpha/c$
Transient EM	Magnetic components	$\mu\epsilon$	$\mu\sigma$
Georadar	Electrical components	$\mu\epsilon$	0
Low-frequency EM	Electromagnetic fields	0	$\mu\sigma$
Geothermal	Thermal fields	0	$1/\kappa$
Electical/Magnetic/Gravity	Potential fields	0	0

Table 2.1: Wave equation coefficients for various geophysical methods

more of the physics within a given imaging algorithm. Moreover, acquisition techniques are constantly improving, meaning datasets are increasing in size - both in terms of the number of shot points and the quantity of offsets/azimuths present. Most of this discussion will be limited to an acoustic, isotropic Earth, but solutions to the two-way wave equation will always be used. The reasons for these assumptions will be discussed at length in the forthcoming chapter.

TWO WAY WAVE-EQUATION IMAGING

The acoustic wave equation can be expressed in equation 2.1.

$$\nabla^2\mathbf{u} - \mathbf{h}\frac{\partial^2\mathbf{u}}{\partial t^2} - \mathbf{g}\frac{\partial\mathbf{u}}{\partial t} = \mathbf{f} \tag{2.1}$$

$$\mathbf{u} = u(x, y, z, t); \mathbf{f} = f(x, y, z, t). \tag{2.2}$$

Here, $\mathbf{u}$ represents the wavefield, t represents time, $\mathbf{h}$ represents the acceleration coefficients, $\mathbf{g}$ represents the diffusion coefficients and $\mathbf{f}$ is a forcing term. Depending on the situation, $\mathbf{h}$ and $\mathbf{g}$ may be scalar or vector terms. Table 2 details six different geophysical scenarios and the values of these differential coefficients under certain assumptions.

Of course, the wavefield of interest herein is the seismic reflection pressure field, and it will be assumed that the diffusion term is negligible ($\mathbf{g} = \mathbf{0}$). This assumption

is valid for the scope of this thesis discussion.

$$\frac{\partial^2 \mathbf{u}}{\partial t^2} = \mathbf{v}^2 \nabla^2 \mathbf{u} \tag{2.3}$$

$$u_{t+1} = -u_{t-1} + 2u_t + \Delta t^2 v^2 \nabla^2 u_t. \tag{2.4}$$

Numerical solutions to these various wave equations have been extensively researched, and implementations include frequency domain propagation (Pratt, 1999), time domain propagation (Boore, 1972), mixed domain techniques (Sirgue et al., 2010), finite difference and finite element techniques (Moczo et al., 2007), and many more. For 3D wave propagation, decomposed across a network of computers, a time-doman finite-difference approach can provide sufficiently accurate, computationally efficient solutions (Komatitsch et al., 2010). The resultant equation to be solved is shown in equation 2.3.

The second equation represents how to practically update the solution. u_{t+1} is the 3D wavefield, at time $t + 1$, being solving for (one time increment in the future), u_t is the current 3D wavefield, u_{t-1} the previous wavefield, $\Delta t^2 v^2$ the time sampling time squared multiplied by the current velocity value squared and, finally, $\nabla^2 u_t$ is the second-order spatial derivative, or Laplacian, of the current 3D wavefield. It is clear, through comparison of the above equation, that these first two terms approximate the time derivative of the wave equation, and the final term approximates the spatial derivative.

Implementing the time derivative is relatively trivial, and a simple subtraction of the two wavefields is acceptable. Although, more accurate options exist (Shubin and Bell, 1987). For the spatial derivative/Laplacian choosing an exact breakdown is more subtle. This depends on the desired accuracy and stability of the solution, and more subtly on the range of velocities present and the desired propagation bandwidth.

It is desirable to use a large stencil to approximate the spatial derivative - the larger the stencil, the more stable the numerical solution, since instabilities can be

implicitly smoothed over (Dablain, 1986). These short wavelength instabilities are of particular concern for exploration seismology, since velocity fields often feature sharp contrasts over short distances. These can be caused by geological scenarios such as salt bodies and anhydrous layers (Mulder, 2001).

This larger stencil also means that the field can have a sparser representation, for a given level of derivative accuracy. A large stencil requires more computation; current industry applications vary from using a 4th order stencil to a 40th order stencil. Generally, the choice depends on how many time-steps the wavefield will be propagated for, as each time step introduces new errors which are carried forward. An 8th order stencil introduces an error of approximately 10^{-7} per time step, this can be estimated by looking at the smallest stencil coefficient (Marfurt, 1984).

A representation of an 8th order stencil can be seen in Figure 2.1. It is comprized of 25 coefficients (nine in each dimension with a shared central value), only five of these coefficients are unique: the stencil is symmetric in all three dimensions about the central value. The Laplacian is applied by convolving the stencil about every point in the current wavefield, and multiplying by $\Delta t^2 v^2$, where Δt^2 is a constant and v is the velocity field value for the central stencil point.

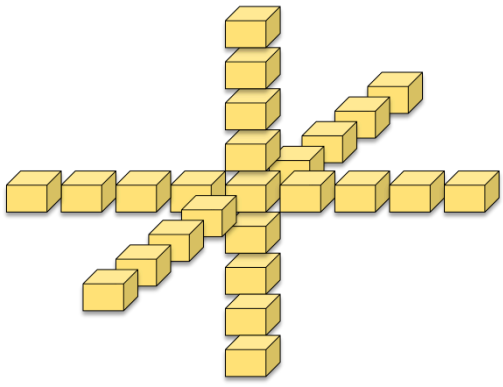

Figure 2.1: Representation of a centred, symmetric, 8th order finite-difference stencil. [**NR**] chap2/. asten

Reverse Time Migration

Simulating the wave-equation provides the core of seismic imaging, but is only part of the process. The act of repositioning reflection events to create a representative Earth image is called migration, and there are many approaches.

Baysal et al. (1983), formally introduced the concept of Reverse Time Migration for exploration seismology, referred to as RTM. It's construction is relatively intuitive: the algorithm propagates both a source wavefield and a receiver wavefield (these will be in 4D, x, y, z and time.) The source wavefield is a forward simulation of the seismic experiment, the receiver wavefield is the back projected data. Events in these fields which are coincident in space and time will represent a true reflection event, all other events are non-physical. By applying an appropriate imaging criterion to these wavefields, a representative image of the Earth scattering field can be created. For RTM, this imaging condition is the cross-correlation of these wavefields, at zero time-lag. This is a representation of Claerbout's imaging condition (Claerbout, 1971), which was conceived by using the exploding reflector model (Claerbout and Black, 2008).

$$m(\mathbf{x}) = \sum_{\mathbf{x}_s,\omega} f(\omega) G_0(\mathbf{x}, \mathbf{x}_s, \omega) \sum_{\mathbf{x}_r} G_0(\mathbf{x}, \mathbf{x}_r, \omega) d^*(\mathbf{x}_r, \mathbf{x}_s, \omega) \tag{2.5}$$

A frequency domain representation of RTM is more compact, and can be seen in equation 2.5. Here, $\mathbf{x}$ represents a generic coordinate system, $\mathbf{x}_s$ the current source wavefield coordinate, $\mathbf{x}_r$ the current receiver wavefield coordinate and ω the temporal frequency. $m(\mathbf{x})$ is the scattering model, $f(\omega)$ the injected source function and G_0 are Green's functions that represent the wavepaths between the relevant coordinates. Finally, the two summation operators approximate time and space integrals, and d^* is the complex conjugate of the recorded data.

The aforementioned imaging condition is the implied multiplication of these two wavefields, and subsequent sum over ω. This is equivalent to integrating over time,

over the source coordinates, and over the receiver coordinates, resulting in an image created in all spatial dimensions. Only events coincident at the same point in time (and space) coherently interfere, to contribute to the estimate of $m(\mathbf{x})$. Other events are incoherent, and will correlate incoherently.

The data term features a complex conjugate, this reverses the sense of time of the receiver wavefield with respect to the source wavefield. This is clear when considering equation 2.5 expanded with the implied Fourier transforms; the conjugate changes the sign of the exponent which reverses the sense of time. Consequently, these wavefields cannot be estimated concurrently. When solving for $m(\mathbf{x})$, it is necessary to propagate and save these 4D wavefields separately. Typically, the source wavefield is computed and saved, the receiver wavefield is then back-propagated and the imaging condition is applied at each imaging time step. The result of this correlation is summed to the estimate of the scattering potential.

Multiple additional methods for dealing with this temporal incongruity have been postulated, and some are more appropriate for certain computational systems than others. These will be discussed in Chapter 6 and Section 6.

While RTM can produce valuable images from few underlying physics assumptions, often, unphysical artifacts and imaging noise are prevalent. This is particularly true if acquisition was not uniform, or strong reflector contrasts were present. To improve the process RTM can be extended to linearized inversion.

LINEARIZED INVERSION

RTM can be considered an adjoint procedure (Tarantola, 1984a). RTM is the adjoint of an idealized modeling operator known as the first-order Born scattering operator. Applying this adjoint will serve to approximate the inverse of the operator, but ideally a better estimate of the inverse is desired (Alewine, 1974).

$$d(\mathbf{x}_r, \mathbf{x}_s, \omega) = \omega^2 \sum_{\mathbf{x}} f(\omega) G_0(\mathbf{x}, \mathbf{x}_s, \omega) m(\mathbf{x}) \sum_{\mathbf{x}} G_0(\mathbf{x}, \mathbf{x}_r, \omega) \tag{2.6}$$

This idealized modeling procedure is represented as equation 2.6, where the terms have the same meanings as in equation 2.5. If this equation describes how the physics links the scattering potential, $m(\mathbf{x})$, to the recorded data, $d(\mathbf{x}_r, \mathbf{x}_s, \omega)$, then it is desirable to "un-do" the physics to recover the model. The exact inverse of equation 2.6 does not exist (the corresponding matrix is both singular and not square), but it is possible to achieve a better approximation than simply applying RTM. One solution is to pose the problem as an inversion.

This has been studied in detail over the last few decades, in reference to improving seismic imaging. Nemeth (1996) initially presented the theory and results for, what they referred to as, Least-Squares Migration. This was posed as a linearized inversion problem, and Kuehl (2002) expanded on this, in the particular focus of wave-equation migration and inversion. Clapp (2005) demonstrated multiple regularization options for improving least-squares imaging, particularly around salt bodies, and Nemeth et al. (1999) extended their orginal work to demonstrate the effectiveness of this technique with incomplete data. In accordance with the nomenclature of these previous studies, the extension of RTM to a linearized inverse process will be referred to as Least-Squares RTM, or LSRTM, from hereon.

$$\mathbf{d} = \mathbf{L}\mathbf{m} \tag{2.7}$$

$$\mathbf{m} \approx \mathbf{L}'\mathbf{d}. \tag{2.8}$$

Initially, equation 2.6 could be simplified by writing the full Born operator as $\mathbf{L}$, the data as $\mathbf{d}$ and the model as $\mathbf{m}$. Implicit in $\mathbf{L}$ is all knowledge of source coordinates, receiver coordinates, and the velocity field. The modeling procedure is now equation 2.7. Accordingly, RTM is now represented more simply as equation 2.8,

where again $\mathbf{L}'$ is the adjoint of $\mathbf{L}$.

This can be considered as a discretized linear system, whereby the linear operator, $\mathbf{L}$, links our recorded observations to the model we are trying to obtain. In the case of seismic reflection imaging these data are recorded in a plane on the surface of the Earth. However, the model to be delineated, $\mathbf{m}$, exists in three dimensions. The resultant system is underdetermined: there is not enough information in these data to uniquely determine the model. From the view of linear algebra, $\mathbf{L}$ contains some zero eigenvalues, meaning it is rank deficient and consequently non-invertible.

$\mathbf{L}$ is not a square matrix, and exhibits a null space. A square matrix that can be estimated is $\mathbf{L}'\mathbf{L}$, and since it is square, $(\mathbf{L}'\mathbf{L})^{-1}$ can be constructed. The matrix $\mathbf{L}'\mathbf{L}$ is often referred to as the Hessian. Thus, it is possible to solve the system in equation 2.9.

$$\mathbf{m} = (\mathbf{L}'\mathbf{L})^{-1}\mathbf{L}'\mathbf{d} \tag{2.9}$$

However, computing the Hessian matrix, let alone the inverse, is not possible due to size constraints. Instead, the process can be posed in a different way. An algorithm can be used that attempts to reduce a measure of how closely the current model estimate matches the data. There are many ways to measure this misfit, and these are referred to as 'norms.' Least-sqaures imaging uses the L2 norm to measure this misfit, this is a scalar measure, and in this context would be the sum of the difference of the squares of the two datasets ($\Sigma(\mathbf{d}_{obs}^2 - \mathbf{d}_{est}^2)$).

Using this scalar measure of data similarity, an objective function can be constructed; this is a function that will aim to either minimize or maximize a numerical value. In this case, the objective will aim to minimise the L2 norm of these datasets, since this measurement will tend to zero as these datasets approach each other. Two such objective functions can be seen in equation 2.10.

$$J(\mathbf{m}) = \|\mathbf{d} - \mathbf{Lm}\|_2^2 \tag{2.10}$$

$$J(\mathbf{m}) = \|\mathbf{d} - \mathbf{Lm}\|_2^2 + \epsilon\|\mathbf{Am}\|_2^2 \tag{2.11}$$

If these data are noise free and the operator exactly describes the physics, then it is possible to reduce $J(\mathbf{m})$ to zero.

Data-space inversion

At this juncture it is necessary to consider two complementary approaches to the problem, these are data-space inversion and model-space inversion. In data-space inversion, forward modeling is used to estimate the misfit between the estimated data and the observed data (Tarantola, 1984b). As a sequential process, consider algorithm 1.

Algorithm 1 Linearized inversion

Calculate initial data-space residual $r = Lm_0 - d$
while iter $<$ n_iter; iter++ **do**
  Create gradient $g = L'r$
  Create conjugate gradient $cg = Lg$
  Calculate step length
  Update m and data residual
end while
Output model

The stopping criterion (when to output the final model) can either be designed to halt after a certain minimum residual is reached, or after a certain computational duration. For inverse imaging it is usually the latter.

This construction assumes that the background velocity is well constrained. Implicitly, Born modeling states that the velocity model can be divided into high-wavenumber and low-wavenumber components (Tang et al., 2009). The former of

these controls the position and amplitude of all reflection events (a representation of the reflectivity), the latter controls the kinematics of the wavefield.

$$\mathbf{m} = \mathbf{b} + \mathbf{h} \tag{2.12}$$

Ordinarily, linearized inversion will be applied once a reasonable background velocity model has been estimated, otherwise events will be be misaligned when comparing estimated and observed data. Alternatively, linearized inversion can be augmented with a velocity update scheme, where imaging is the 'outer loop' of the process, and the 'inner loop' acts to update the velocity model (Symes and Carazzone, 1991). Velocity update schemes will not be discussed in detail in this thesis, but their importance should be mentioned while discussing seismic inversion.

SEAM EARTH MODEL

Artificial Earth models can be invaluable for understanding how these imaging systems work. To understand and quantify how RTM and linearized inversion contrast it is necessary to use a representative example, where the physics can be controlled. To test these algorithms an imaging target that contains a wide variety of wavenumbers and velocity contrasts is desirable, in this section part of the SEAM (Vigh et al., 2009) model is used. A constant density assumption will be implicit during these upcoming sections.

Figure 2.2 shows a section of the model which was chosen. Features include continuous sedimentary reflectors, rugose and discontinuous reflectors, and a high contrast, steeply dipping salt body. Within this salt body are sedimentary inclusions and a carbonate top. An effective imaging algorithm will be able to resolve these features and converge towards the correct image, within given acquisition restrictions. A high wavenumber representation is shown in Figure 2.3: these are the features that the imaging methodology should aim to recreate.

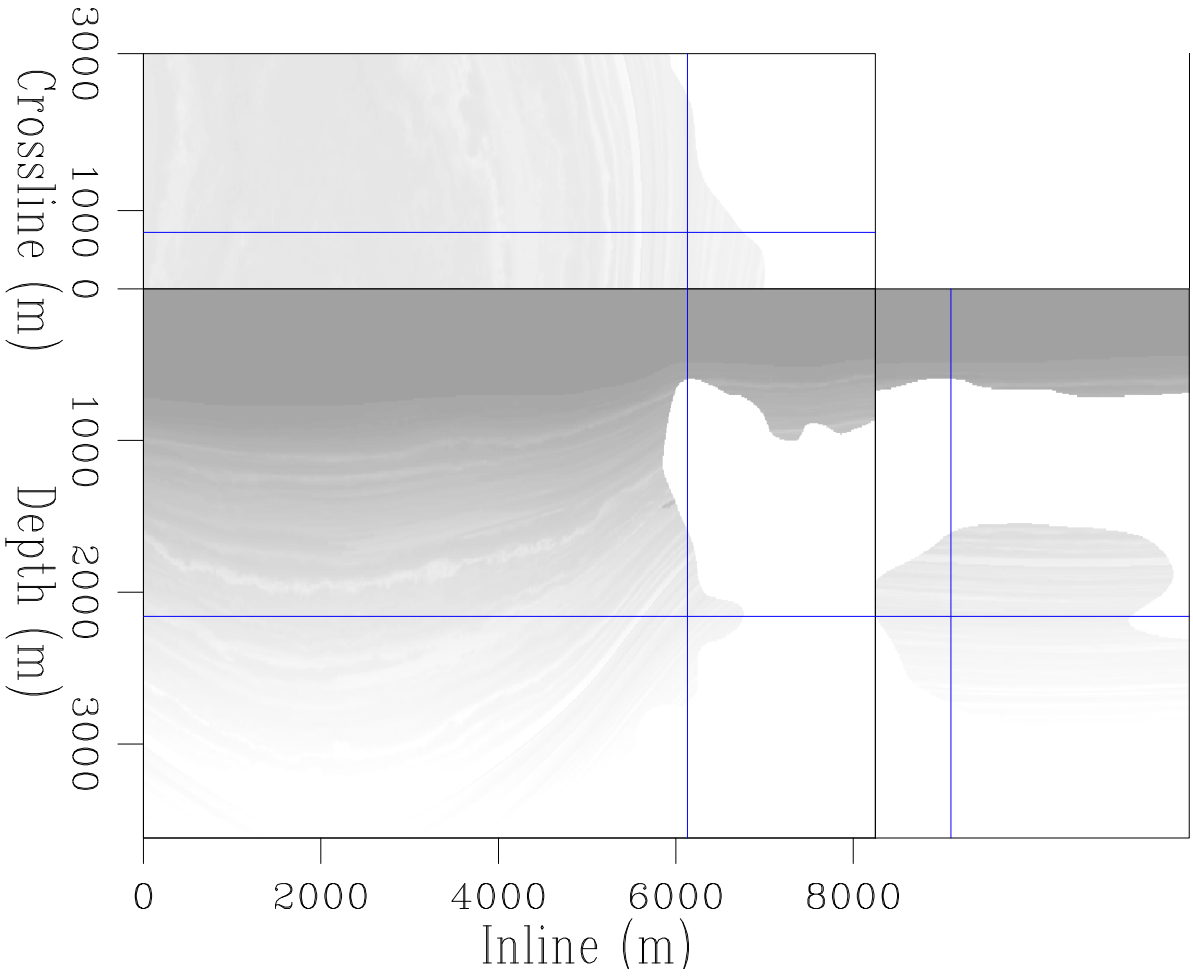

Figure 2.2: Section of the SEAM velocity model, used for modeling and imaging. [**ER**] chap2/. seamsmall

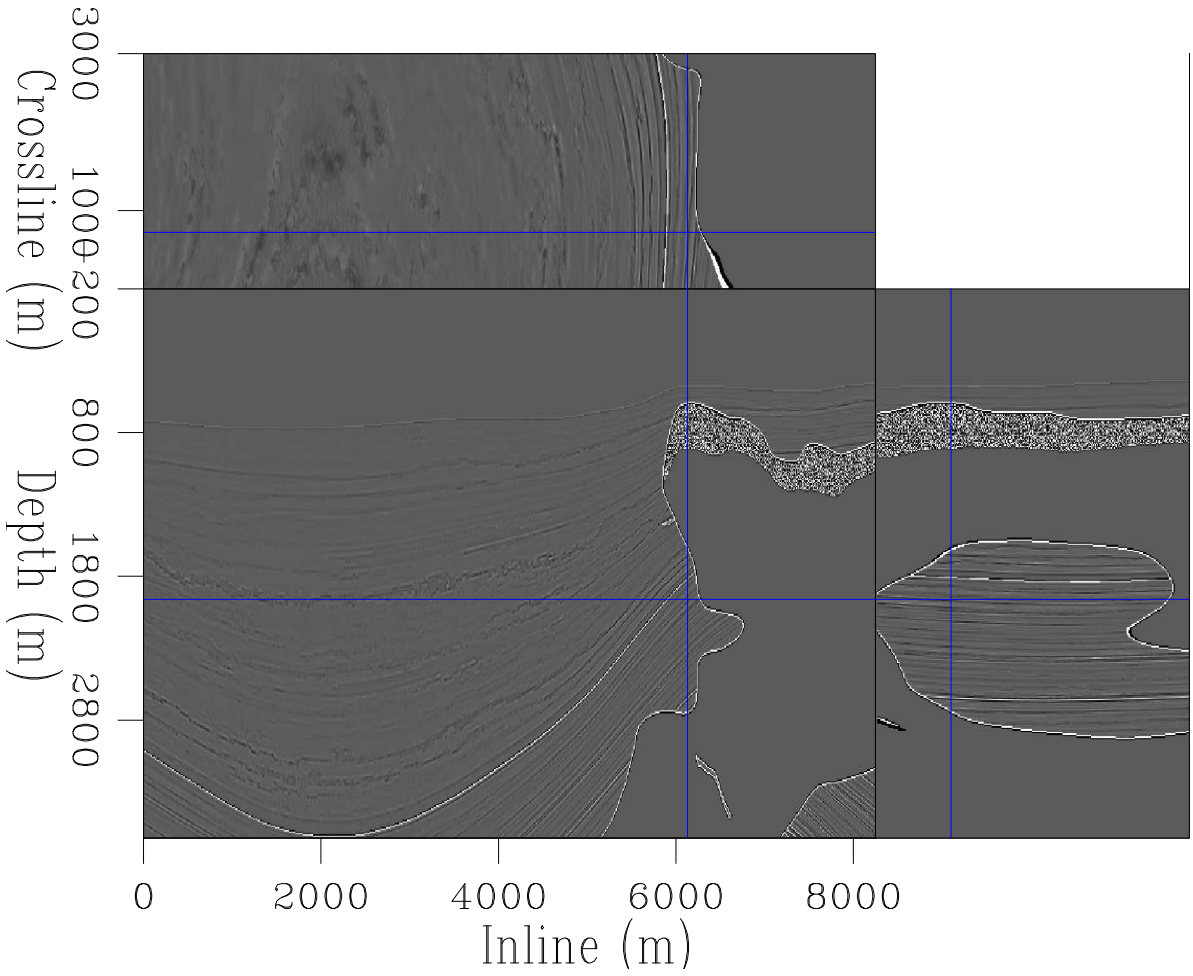

Figure 2.3: The scattering potential to be imaged, calculated by subtracting a lightly smooth version of the velocity model from itself. [**ER**] chap2/. seamsmallrefl

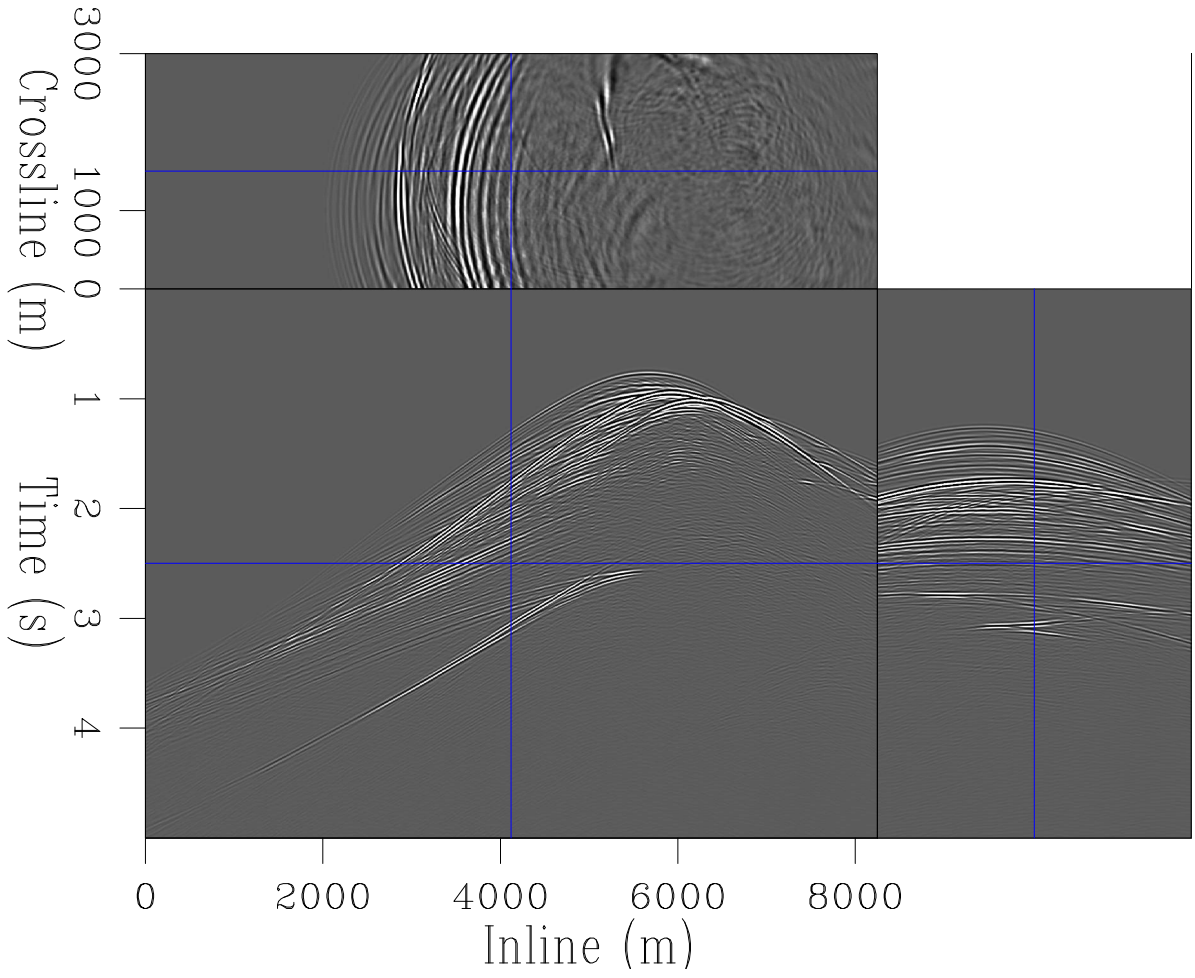

Figure 2.4: An example of a 3D shot simulated using linearized forward modeling over the proposed velocity model. [**CR**] chap2/. seamsmallborndata

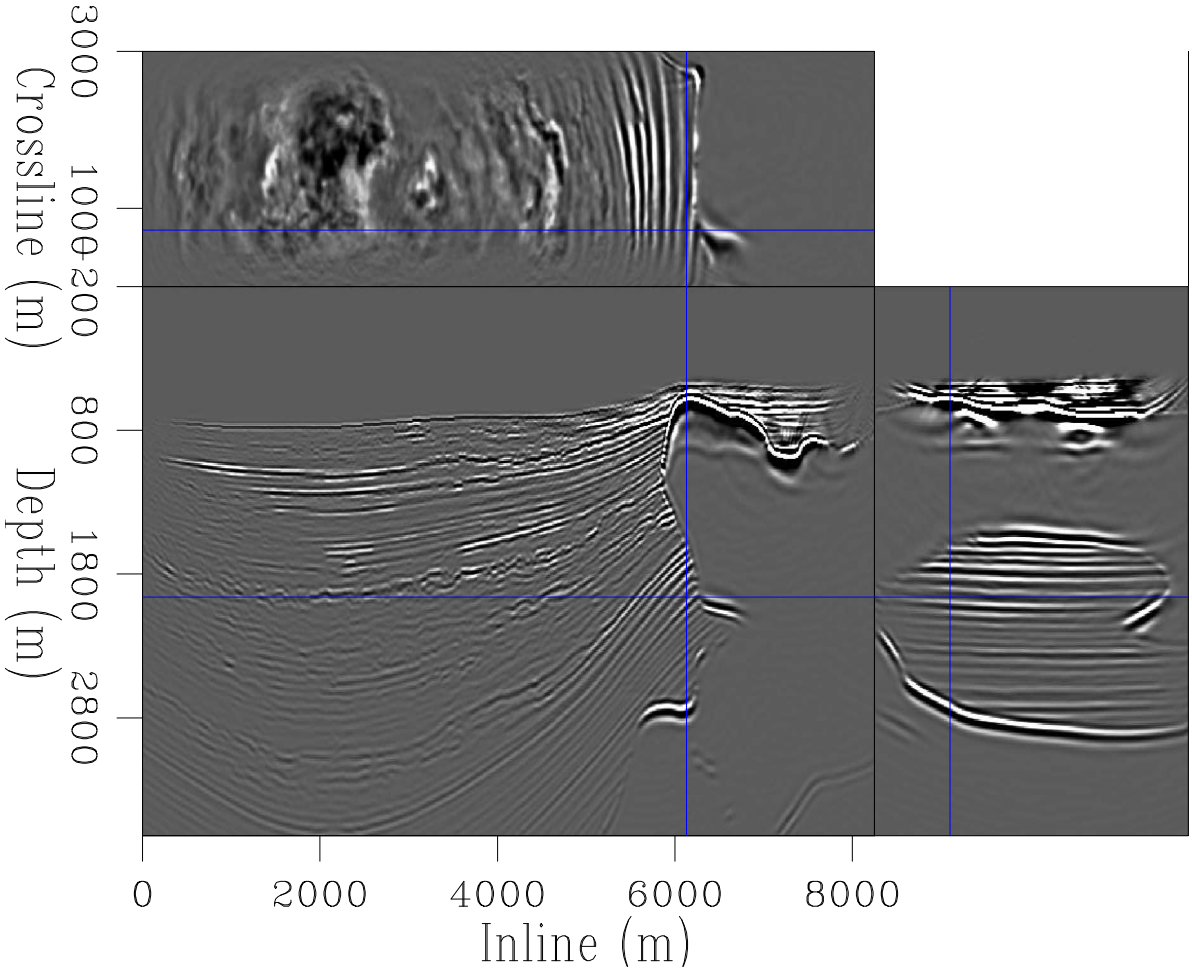

Figure 2.5: The unfiltered RTM image produced from the synthetic data shown above. For this image the source domain was well sampled with 800 inline shots and 30 crossline shots. [**CR**] chap2/. seamsmallimg

An example of a 3D shot simulated over the model is shown in Figure 2.4. The geometry was a relatively sparse, fixed receiver type survey to emulate that of Ocean Bottom Node (OBN) acquisition. Sources were at the surface in a regular grid, receivers were on a sparse grid below the surface with dense inline sampling (35 receivers, one every 200 m) and sparse crossline sampling (two inlines separated by one kilometre.) The Earth model itself is sampled every 10 m in the three cartesian directions and extends to a depth of 3600 m (almost 12,000 ft), has an average water depth of 800 m (2600ft) and extends 8km x 3km in the x and y directions respectively (26,000 ft x 10,000 ft). To mitigate spurious reflections from the domain boundaries during propagation each dimension is padded by an additional 40 model points (another 400 m, or 1300 ft.)

To create these data Born modeling was used. For imaging RTM is used first; this was a full 3D algorithm using random boundaries for the source wavefield modeling, for more details see Chapter 6. For the random boundaries the same additional padding was used (40 model points.) The image obtained from RTM is shown in Figure 2.5. Noticeably, many nuances of the high wavenumber model have been successfully recreated. Upon some scrutiny, several shortcomings are identifiable. Amplitudes are incorrectly balanced, the steep edges of the salt body are not well focused, the general frequency content of the image is lower than the model (not well balanced through the wavenumber domain) and low frequency artifacts are prevalent throughout. There is also an imprint of the wavelet along the reflectors.

These image imperfections can be attributed to two origins: the imaging procedure was not a true representation of the underlying physics (Yoon et al., 2003); the acquisition was limited to a surface plane (Pan et al., 1988). Image noise is induced from limited acquisition, wavelet effects, random correlation noise and from coherent correlation noise (Guitton et al., 2006). These will be briefly elaborated upon, in the next few paragraphs.

The imaging condition (equation 2.5) will result in a squaring of the source wavelet, which is implicit in both f and d. Thus, reflectors will appear more spread out, as their spectra will be shifted; this is the wavelet effect alluded to. High frequency noise

is present in the image (largely) because of random boundary source wavefield propagation, although this noise source is not significant. More noticeably, low frequency, unphysical events are seen as a result of coherent correlation noise. The imaging condition only holds for wavefields traveling in opposite directions. In a complex velocity model, occasionally the source and data wavefields may travel in the same direction. When correlated, low-frequency artifacts will be created (Fletcher et al., 2005). In particular, this is seen around the edges of the salt body (Jones and Davison, 2014). It is possible to harness this effect for velocity updates, since this same direction correlation can provide tomographic information (Woodward, 1992).

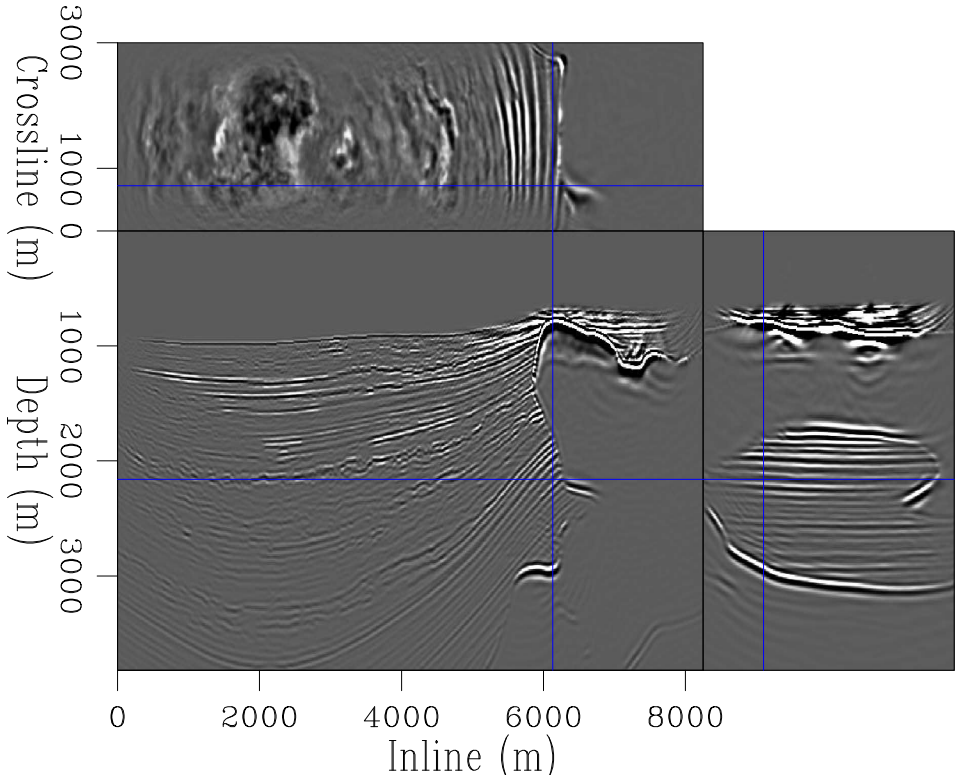

Figure 2.6: The recovered scattering model after a single iteration of conventional linearized inversion. [**CR**] chap2/. seamsmallinv1

Figure 2.6 shows the same image crosss-section after a single iteration of linearized inversion, then Figure 2.7 shows the same section after ten iterations of linearized inversion. Each iteration of linearized inversion is twice the cost of a single pass

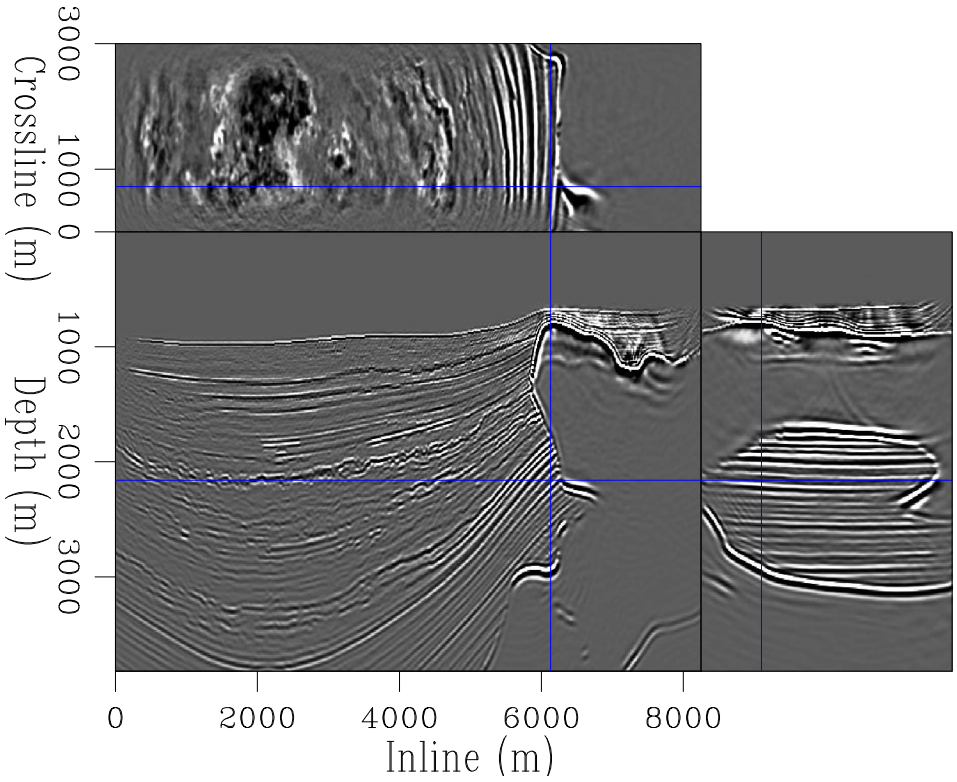

Figure 2.7: The recovered scattering model after ten iterations of conventional linearized inversion, clipped identically to Figure 2.6. [**CR**] chap2/. seamsmallinv10

of RTM imaging. A conjugate direction algorithm, (Fox, 1971), was used at each iteration.

It is clear that this process improves upon all these imaging imperfections. Frequency content is higher and more consistently balanced, low-wavenumber artifacts are entirely gone and reflector amplitudes (and illumination) are representatively balanced throughout the section. In particular, inclusions in the salt body are more pronounced and the low-contrast carbonate atop the salt has been resolved. Additionally, the rugose nature of many of the shallow reflectors has been well resolved.

To measure how the procedure is performing the objective function can be plotted as a function of iteration number. Figure 2.8 shows this, by analysing at the normalized data-space residual (taking the difference of the squares of the estimated data and the observed data.) If the residual is zero then the estimated data exactly matches the observed data. After ten iterations the system has converged to within a few percent. This is as expected, since noise-free, Born modeled data were used. With field data convergence to within ten, or even twenty, percent is regarded as successful.

PRECONDITIONING

For many applications, even ten iterations can become prohibitively expensive. The results in the previous section took the most naive approach to linearized inversion, with no attempt to influence the system with any prior knowledge. There are two main categories of influence that can be applied in seismic inversion, known as regularization and preconditioning (Claerbout, 2001). The former acts to steer the model space towards some preconceived notion or statistic that the model should exhibit (Fomel, 2001), the latter aims to create a change of model variable that may improve convergence characteristics (Steihaug, 1983). This section will focus on preconditioning, whereas Chapter 4 will address some regularization options.

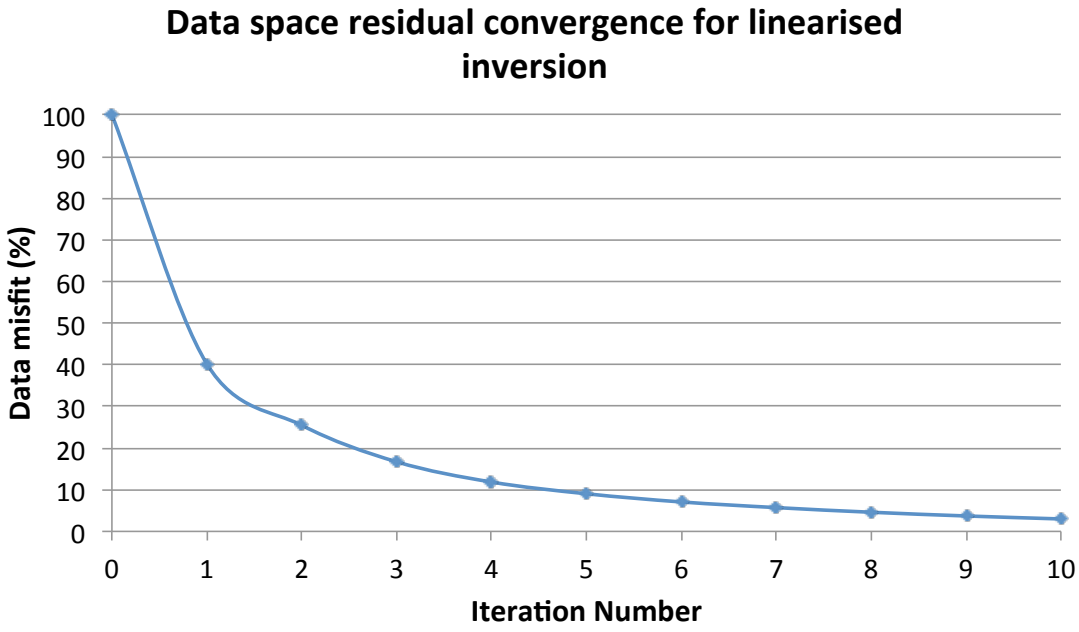

Figure 2.8: Normalized data misfit as a function of iteration number. Measured from the SEAM model above. [**NR**] chap2/. seamsmallliconv

$$J(\mathbf{m}) = \|\mathbf{d} - \mathbf{Lm}\|_2^2 \tag{2.13}$$
$$J(\mathbf{p}) = \|\mathbf{d} - \mathbf{LSp}\|_2^2 \tag{2.14}$$

By making the change of variable $\mathbf{m} = \mathbf{Sp}$, equation 2.10 can be rewritten as equation 2.13. During each new iteration this operator, $\mathbf{S}$, and it's adjoint must be applied.

$$\begin{aligned} \mathbf{r} &= \mathbf{Lm}_0 - \mathbf{d} \\ \mathbf{gg} &= \mathbf{L}'\mathbf{d} \\ \mathbf{rr} &= \mathbf{Lgg} \\ \mathbf{r} &= \mathbf{r} + \alpha\mathbf{rr} \\ \mathbf{m} &= \mathbf{m} + \alpha\mathbf{gg} \end{aligned} \tag{2.15}$$

The system will act to recover the most representative $\mathbf{p}$; during the final step this preconditioner can be reapplied to recover $\mathbf{m} = \mathbf{Sp}$.

Equation(s) 2.15 demonstrates the sequence of operations for linearized inversion. By including a preconditioner, the sequence subtly changes to that seen in equation 2.16.

$$\begin{aligned} \mathbf{q} &= \mathbf{LSp}_0 - \mathbf{d} \\ \mathbf{ss} &= \mathbf{S}'\mathbf{L}'\mathbf{d} \\ \mathbf{qq} &= \mathbf{LSss} \\ \mathbf{q} &= \mathbf{q} + \alpha\mathbf{qq} \\ \mathbf{p} &= \mathbf{p} + \alpha\mathbf{ss} \end{aligned} \tag{2.16}$$

It is necessary to consider what the preconditioning operator, **S**, could be to improve convergence. The mathematics of inversion instructs that the gradient can be multiplied by any positive definite matrix and convergence will be observed. It is, perhaps, most instructive to consider how the first gradient could be improved/cleaned to better represent the system. In this case the initial gradient is the RTM image. The subsequent step of the inversion is to model a dataset using this gradient, to construct the conjugate gradient. One could conjecture that a preconditioner that could improve the frequency content of the image and remove the low amplitude artifacts would improve convergence. Options for this could include a vertical derivative, or a spatial Laplacian, applied to the image. Both of these would remove low-wavenumber artifacts and shift the frequency content.

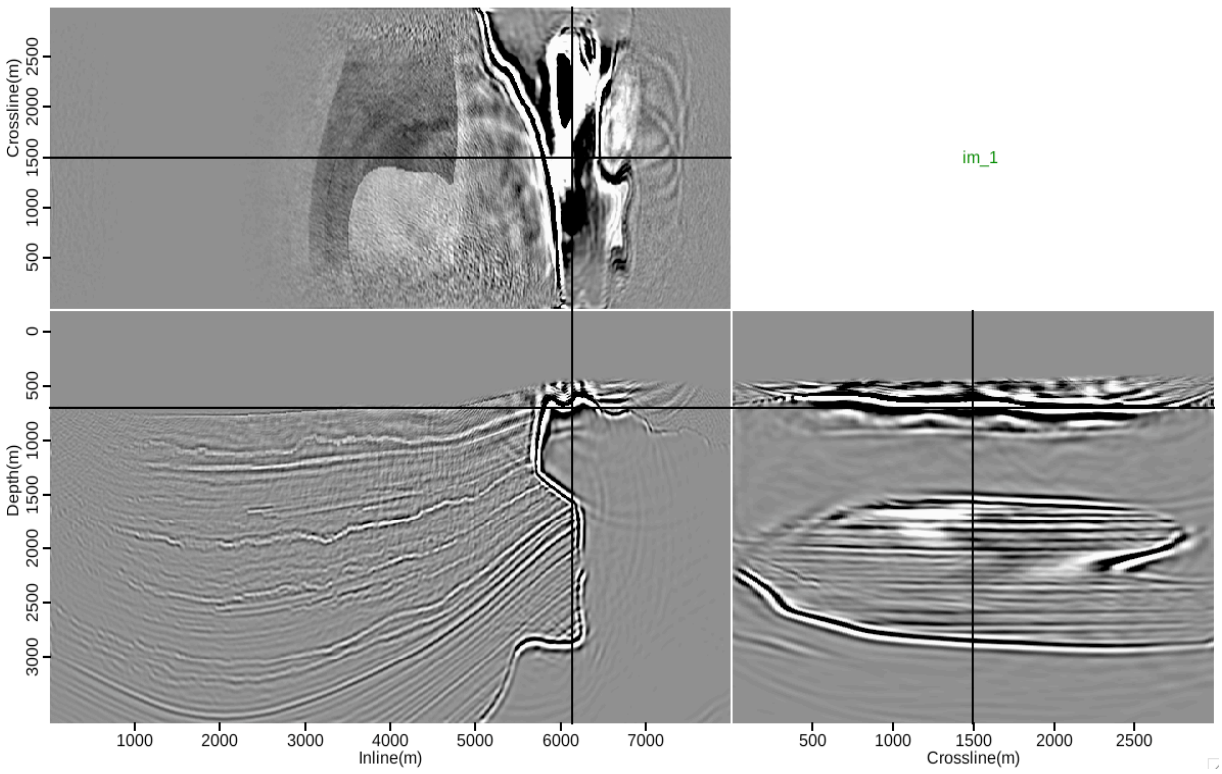

Figure 2.9: The first gradient of linearized inversion without any preconditioning. [**CR**] chap2/. precident

The appearance of the initial gradient after application of these operators can be

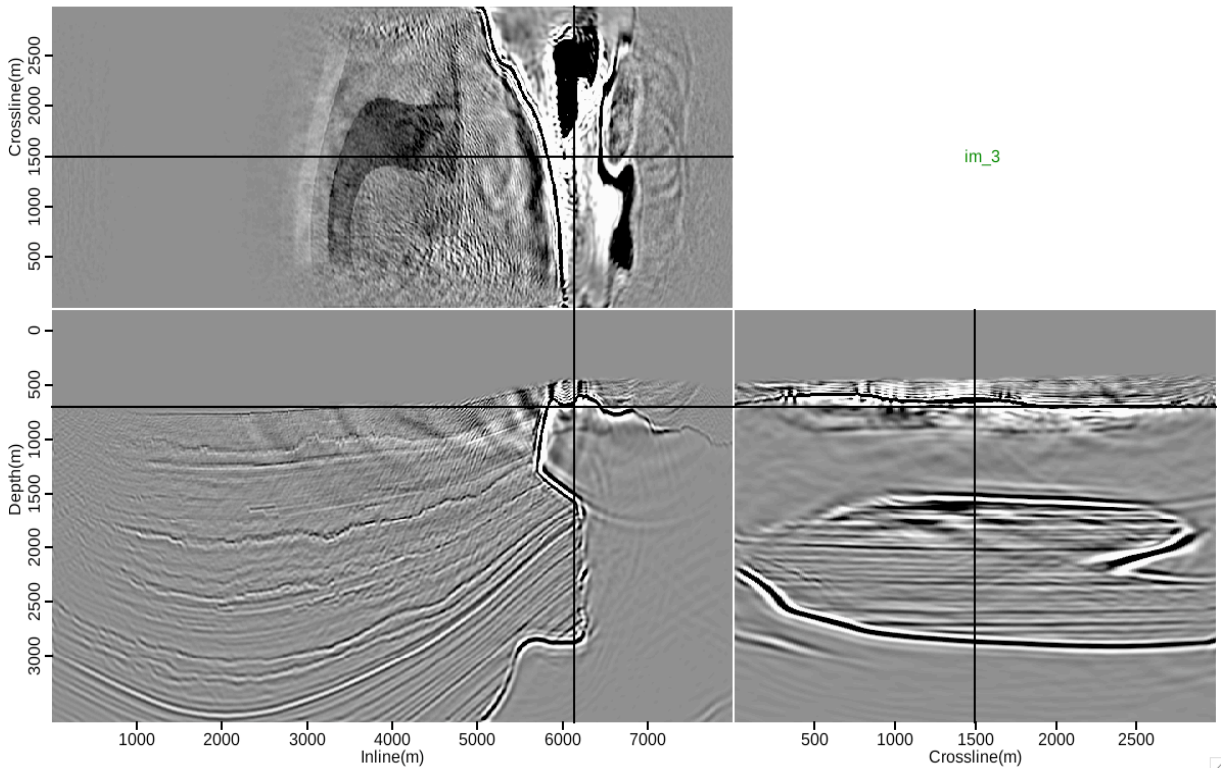

Figure 2.10: The first gradient of linearized inversion preconditioned with a vertical derivative. [**CR**] chap2/. precvd

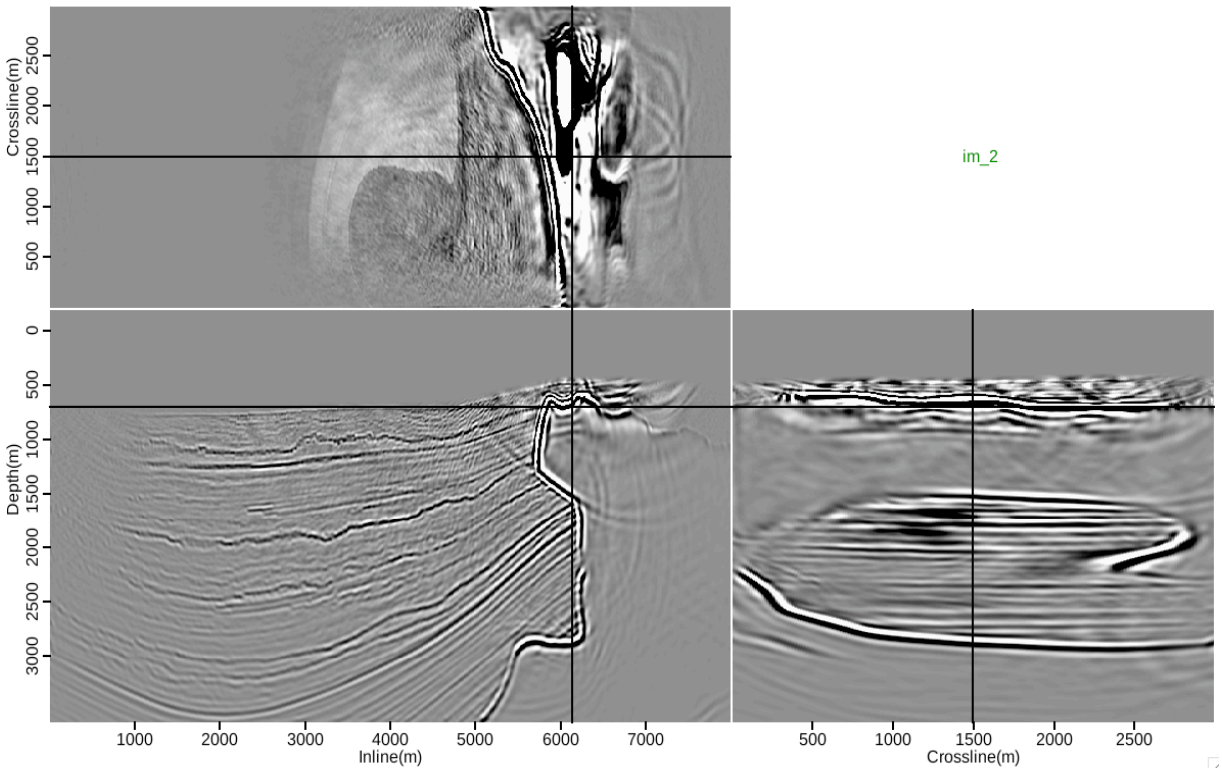

Figure 2.11: The first gradient of linearized inversion preconditioned with a Laplacian. [**CR**] chap2/. preclaplac

seen in Figure 2.9, Figure 2.10 and Figure 2.11. Both operators perform as expected, however, it is important to notice that the vertical derivative sets all zero-wavenumber events to zero, meaning it will destroy any vertical events. This is noticeable on the edges of the salt body.

Convergence curves for three preconditioners (also the identity matrix is used, meaning no preconditioning) can be seen in Figure 2.12. The application of these preconditioners (in terms of computation) is entirely negligible, so the cost of these systems is the same as unpreconditioned inversion. Consequently, there is no need to distinguish between iteration number and cost. In Chapter 3 a wider of inversion techniques will be discussed, and here a distinction between cost and iteration number with become imperative.

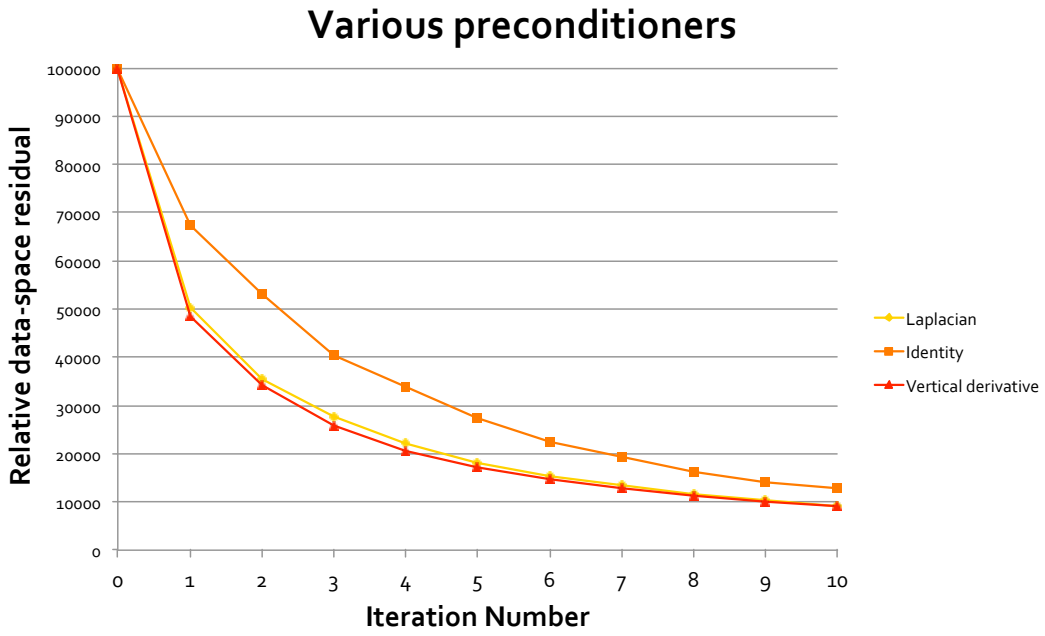

Figure 2.12: How each preconditioned system performs as a function of iteration, measured by L2 data misfit. [**NR**] chap2/. preccurve

Predictably, the preconditioned systems converges to a lower data-space residual faster than the unconditioned system. Particularly for low iteration results the difference is considerable, giving much fast data-space convergence, at no extra cost. In

this case preconditioning is only necessary for the first iteration, as after these artifacts have been removed the inverse system can work on amplitude balancing. One way to think about this is simply providing the system with a more accurate initial model.

Whilst convergence has been accelerated, the procedure it is still very expensive. The subsequent chapter will discuss methods of reducing the data-space in order to greatly reduce convergence as a function of cost.

CONCLUSIONS

Methods of creates seismic images using wave-equation based approaches were discussed. By assuming data were created from an acoustic, isotropic, Earth, the two-way wave-equation can be reduced to three terms - a forcing term, a spatial derivative, and a temporal derivative. If knowledge of the subsurface velocity exists then the seismic experiment can be simulated.

Through simulating the source wavefield, and back projecting the receiver wavefield using the recorded data, RTM can be applied. RTM gives accurate, representative images of the subsurface, within these restrictions. These images can be greatly improved by extending this linear system to an inversion, and by using an update scheme to iteratively improve the model estimate.

This linearized inversion concept, as a function of convergence, can be accelerated by using preconditioners which can remove RTM induced artifacts faster than the inverse scheme.

Chapter 3

Phase-encoded inversion and randomized sampling

Applying linearized inversion to many thousands of shots can quickly become unfeasible; it is possible to intelligently reduce the data size by using two classes of techniques - phase encoding and randomized sampling. It is also possible to augment these two concepts to further increase inversion feasibility. This chapter will address this data size problem, and analyze these solutions.

PHASE ENCODING

Typically, in seismic imaging the size of the data space is much larger than the model space, thus data movement can become an impediment to an efficient inversion scheme. Phase encoding is a method of artificially combining (blending) data that will result in a algorithm that favours additional computation, per iteration, over reduced data movement. In some cases, the overall computation can be reduced. The aim is to macroscopically improve convergence as a function of overall cost.

In phase encoding seismic shots are scaled, shifted and summed together to create a series of super-shots (Morton and Ober (1998); Dai et al. (2012)). In the most

extreme case all shots can be combined to create a single supershot. For Ocean Bottom Node (OBN) or Ocean Bottom Cable (OBC) type surveys this can be viable, since often the receivers are fixed. How many super-shots one should create is a question of survey geometry and source sampling (Ober et al., 2000).

As discussed in the previous and final chapters, data movement can be a significant source of cost during conventional migration and modeling (Neelamani et al., 2010). Every shot must be read and written, from disk, during every iteration. This can create a bottleneck both within and across iterations. Furthermore, as discussed in the final chapter, a GPU based propagator is used. This greatly accelerates the speed of wave propagation, exaggerating the penalties of I/O and disk access. Phase encoding reduces the data-space dimensionality within each iteration, favouring additional computation over disk access. Under many conditions this can give a significant reduction of cost for a given level of sub-surface property convergence. Again, this discussion will focus on scattering potential recovery.

Designing the encoding matrix

The first decision is how to combine these data into subsets. Consider $\mathbf{d}_e = \alpha\mathbf{d}$, where $\mathbf{d}$ is the full data vector, $\mathbf{d}_e$ is a supershot (or series of supershots) and α is the encoding matrix. This encoding matrix contains all information about how to shift, scale and sum the shots together.

Designing α is both a question of convergence properties, efficiency, and data geometry. Phase encoding, for the purposes of seismic imaging, was first postulated in Romero et al. (1999), in the context of combining shots for prestack migration. Herein it is concluded that using a random series of time shifts provides the best results. Furthermore it is claimed that no more then ten shots should be combined at a time. However, if post-imaging amplitude analysis is desirable such a technique can have drawbacks (Zhang et al., 2007).

For pre-stack imaging a variety of different encoding functions, α, have been analysed, in addition to simple random shifts (Godwin et al., 2010). For example, plane

wave encoding, Liu et al. (2002), can perform very well for dense source sampling, since crosstalk artifacts can be mitigated. In the context of phase encoded inversion, as opposed to migration, many more shots can be combined since the solver searches for common model properties between iterations. Krebs et al. (2009) used a single bit encoding function (1s and -1s) to scale the data, performing no time shifts. This results in the most efficient convergence, since the time records are not being prolongated and the cost of each iteration is massively smaller. Many iterations are required to reduce artifacts, thus relative usefulness is determined as a function of cost and convergence.

Phase encoded inversion

Linearized inversion can be extended to incorporate phase encoding. The most natural extension will be termed 'static' phase encoding, whereby the encoding function, α, is constant across iterations. Under this condition the main loop of the algorithm remains unchanged, but in initialisation the encoding is applied to the full dataset to create these reduced data, $\mathbf{d}_e$. Each loop is significantly faster to implement since only super-shots are being looped over, rather than each individual shot. Solver issues aside, this speed up will be directly proportional to the ratio of the total number of shots to the number of super-shots, $n_{shots}/n_{supershots}$. This is largely similar to linearized inversion and is summarized in algorithm 2.

It is immediately apparent that such a scheme will induce many artifacts; moreover, these artifacts will be consistent between iterations. Consequently, many iterations will be required to reach a given level of convergence. Given this, a comparison between phase encoded inversion and linearized inversion will be done under the context of data-residual reduction as a function of computational cost. This cost will be normalized to one full iteration of linearized inversion and convergence will be a normalized measure of the data space residual.

Reducing the cross-talk artifacts in this way can be a slow process. These artifacts are reduced significantly more quickly if the encoding function is changed between

Algorithm 2 Static phase encoding

```
Create α
d_e = αd
while iter < n_iter; iter++ do
  Create gradient of d_e
  Create conjugate gradient (size of d_e)
  Model update
end while
Output model
```

iterations. This will be termed 'dynamic' phase encoding (Boonyasiriwat et al., 2010). This creates a huge advantage for cross-talk reduction: these artifacts will now be incoherent between iterations, causing them to stack-out rapidly. The solver will search for common model components between iterations and thus residual reduction, especially for early iterations, will be accelerated.

The caveat with dynamic phase encoding, however, is that by changing the encoding between iterations the 'observed data,' that each loop is implicitly considering, are changed. Thus the solver cannot update the data-space residual, which is needed to calculate the subsequent gradient. This process is now non-linear, and before the gradient calculation an additional forward modeling step is needed to estimate the updated data-space residual. The consequence of these attributes is that a non-linear solver must be used, and the cost (per iteration) is roughly 1.5x that of the static algorithm. The dynamic sequence is summarized in algorithm 3.

Algorithm 3 Dynamic phase encoding

```
while iter < n_iter; iter++ do
  Create α
  d_e = αd
  Forward model to estimate d_e residual
  Create gradient of d_e
  Create conjugate gradient (size of d_e)
  Model update
end while
Output model
```

The relative advantage of phase encoded inversion is tightly tied to survey type and geometry. Under certain circumstances, the advantage over linearized inversion can be prodigious, under other conditions it can be detrimental. These will be considered for a variety of marine conditions.

For a streamer type geometry, phase-encoding causes two issues. Firstly, the aperture of each supershot is increased as a function of the number of combined shots. The receivers are constantly moving, and each must be considered when forward modeling. As a result, for streamer surveys, the choice of how many shots to combine, or indeed whether phase encoding is advantageous at all, can be hard to determine. Secondly, since the receivers are not static, some shots will not have the full complement of recorded data in order to calculate residuals. This can thought of as: when combining shots together, not each shot is present in each receiver.

For Ocean Bottom Cable (OBC) or Ocean Bottom Node (OBN) type surveys the receivers are spatially fixed (at least for a group of shots). Subsequently, for each combined shot there may be no aperture increase. For inversion, under the most extreme condition, all shots can be summed into one supershot. Also, each receiver has the full complement of shots. The cost of migrating this single supershot will be the same cost as migrating any of the individual shots.

Initially, this latter type of survey will be analyzed. A section of the SEAM model will be used again, with fixed receivers and source positions. Using synthetic modeling 120 shots were simulated over the same model section in the previous chapter and the same linearized results from that chapter will be used for comparison. To push the algorithm, and to reduce data movement as much as possible, all shots will be scaled by a randomly selected single bit function and combined into a single super shot, with no time shifts. Dynamic phase-encoding will be the method used, and this will create a scheme that maximally favors computation over data movement.

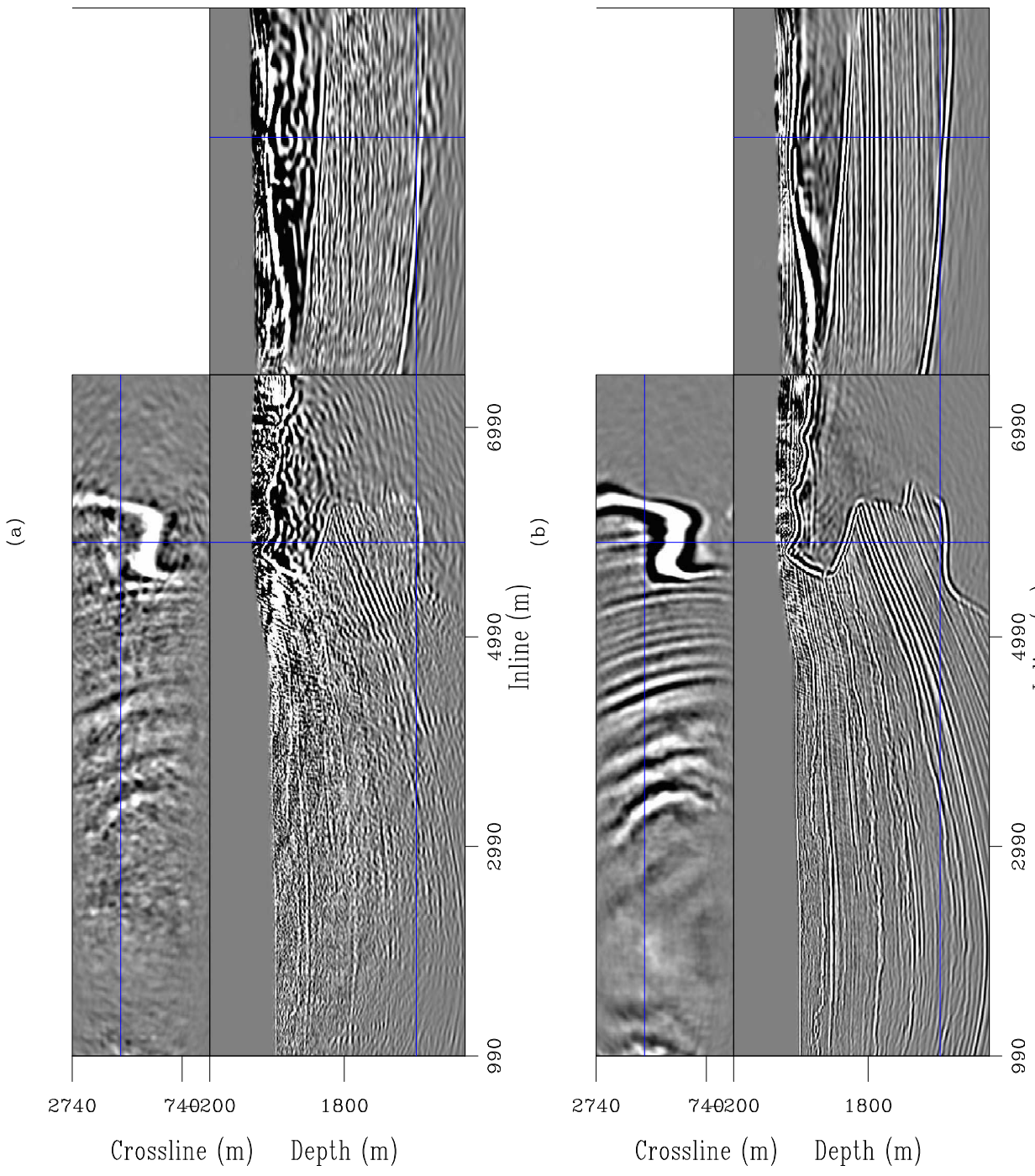

Figure 3.1: The result after one iteration of phase encoded inversion, (a), and after 50 iterations, (b). Here 120 shots were combined into one supershot. [**CR**] chap3/. enc1v50

Unsurprisingly, early images for this choice of data combination are noisy, although macroscopic structure is discernible, and many keys reflectors can be mapped. A comparison between the first iteration image and the 50_{th} can be seen in Figure 3.1. Figure 3.2 then shows the first iteration image from LSRTM compared to the equivalent cost image from phase encoding. It is immediately apparent that the image from phase encoding has improved amplitude balancing, features higher wavenumber content, a reduced acquisition footprint, and an increase in high wavenumber artifacts. Phase encoding preserves all dips present in the data, however it noticeable (particularly in the top panel) that resolution is lost towards the edge of the model, where data redundancy is reduced. This is fairly intuitive, since the phase encoding scheme is more reliant on stacking for coherency.

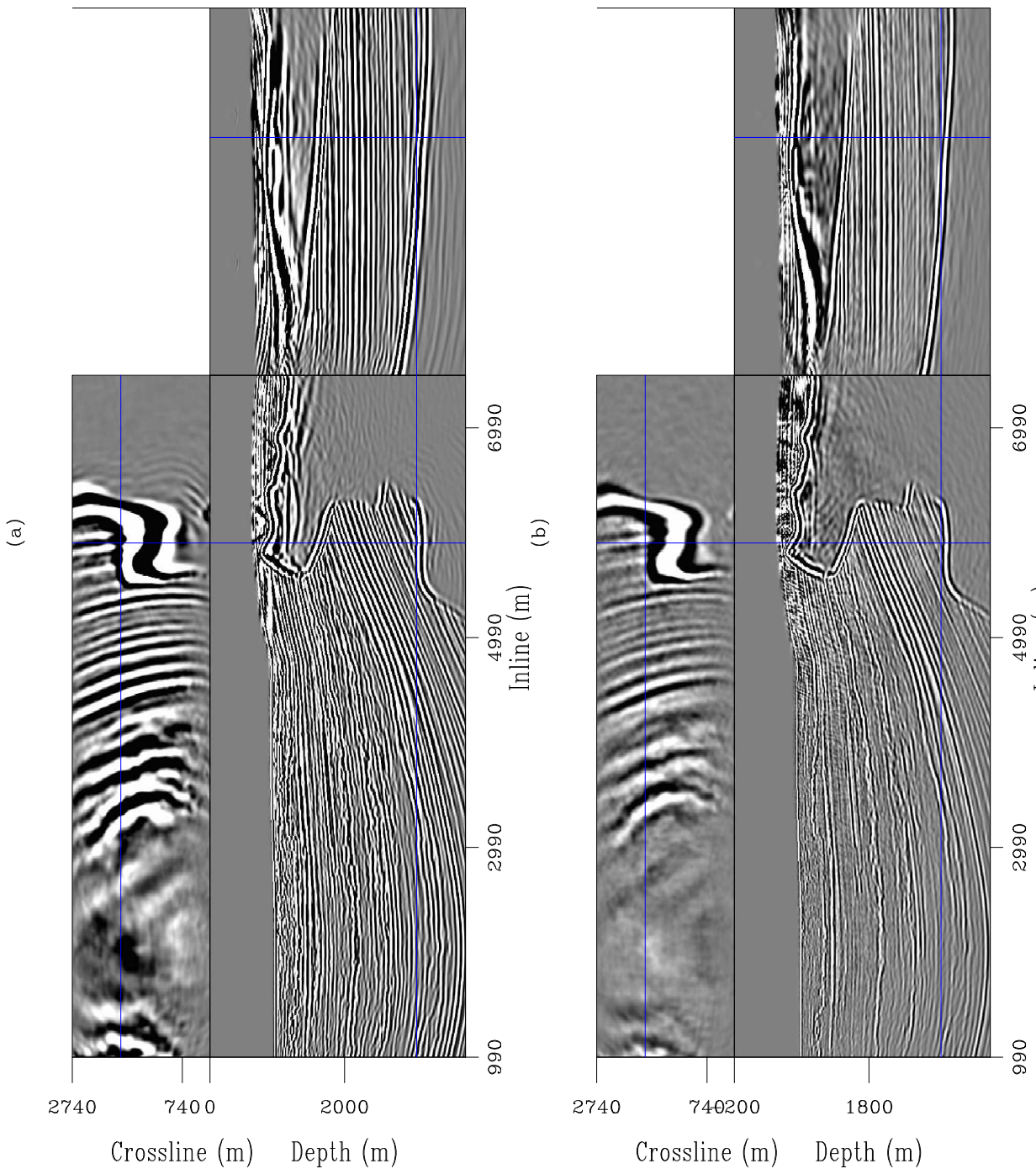

Figure 3.2: The result after one iteration of LSRTM, (a), and after 50 iterations of phase encoded inversion, (b). [**CR**] chap3/. li1venc50

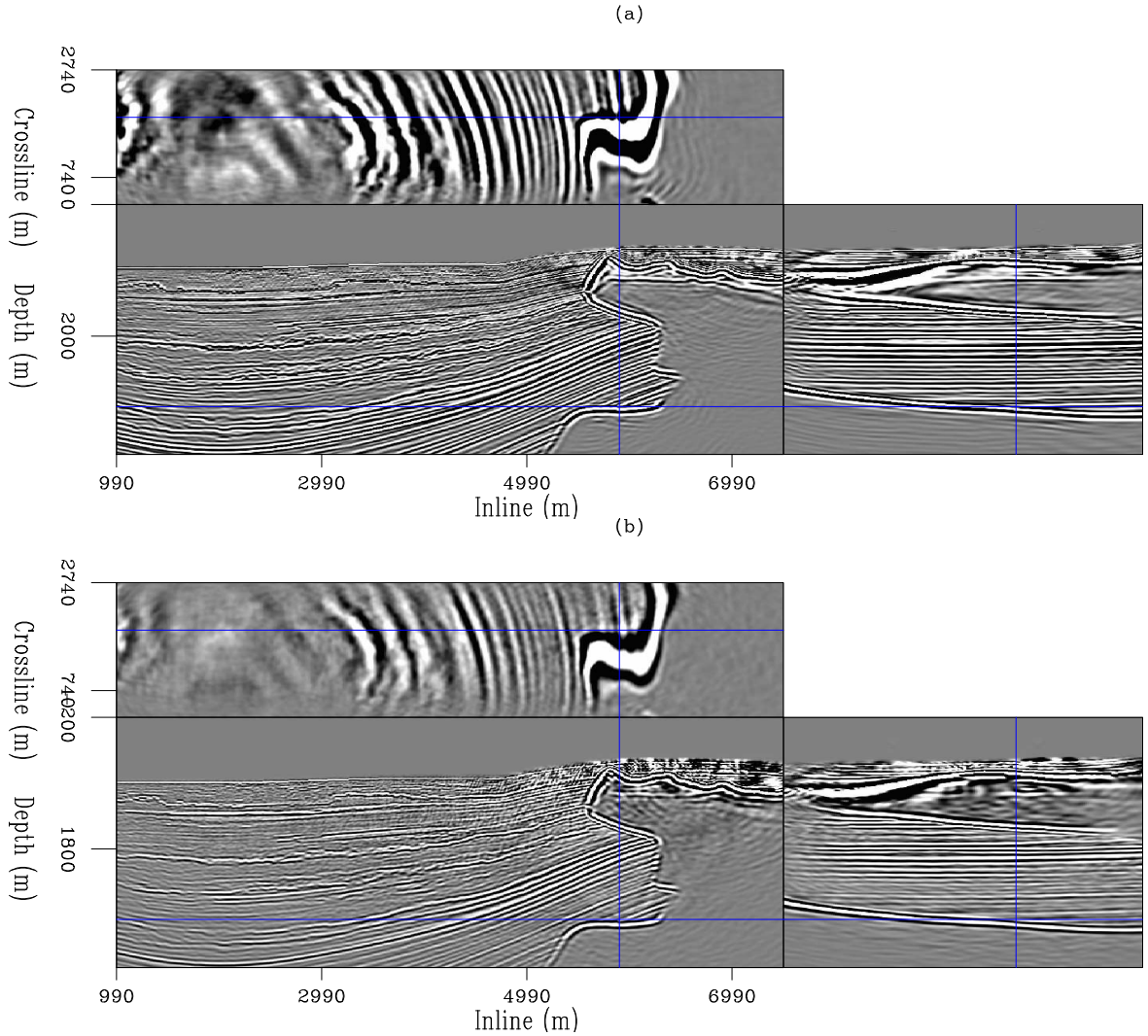

Figure 3.3: The result after ten iterations of LSRTM, (a), and after 50 iterations of phase encoded inversion, (b). [**CR**] chap3/. li10venc50

Next, Figure 3.3 shows a comparison of ten iterations of LSRTM against the equivalent cost image of phase encoded inversion. At this stage the amplitude and wavenumber fidelity of the LSRTM image is much improved, and the relative differences between the images to harder to ascertain. This makes sense by comparing the relative convergence properties as a function of cost, Figure 3.4 and Figure 3.5. This shows that phase encoding falls to a lower residual quickly, but can then plateau. As a function of iteration number, phase encoding is less favourable (in terms of data space residual reduction.) However, when this is posed as a cost comparison, phase encoding can exhibit significant residual reduction before LSRTM has completed a single iteration. A viable strategy would be to use phase encoding to estimate an initial model, and then perform several iterations of LSRTM to reduce noise and represent the amplitude character more representatively.

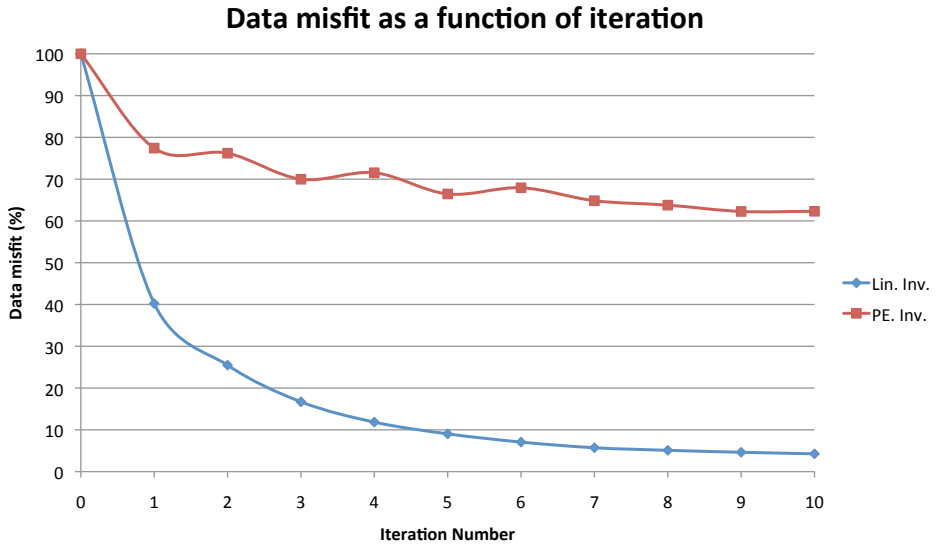

Figure 3.4: How the data fit improves as a function of iteration number, for linearized inversion and for phase encoded inversion. [**NR**] chap3/. convwiter

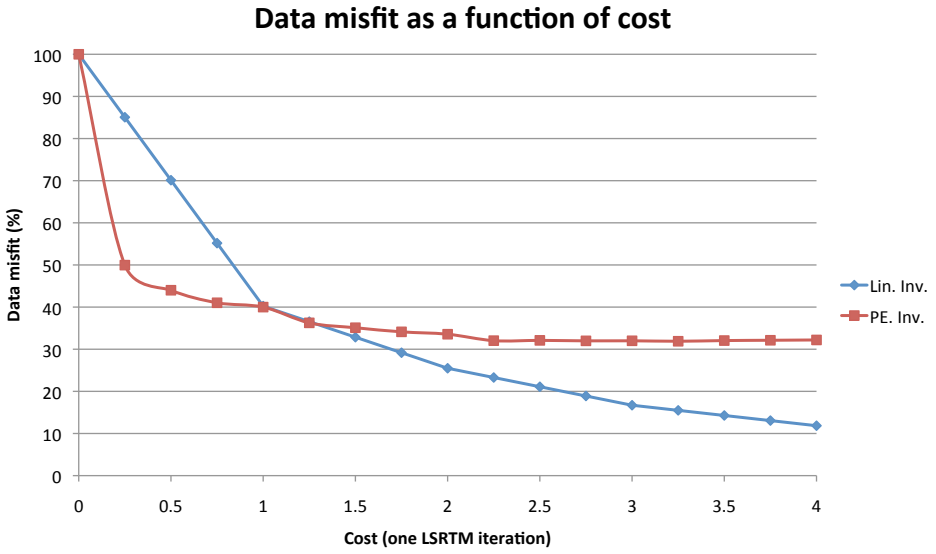

Figure 3.5: The same plot as Figure 3.4, but as a function of cost. The cost is normalized to that of one iteration of LSRTM, with results for LSRTM linearly interpolated for the sake of representation. [**NR**] chap3/. convwcost

Parallel data encoding

Although within each iteration data movement is massively reduced, however, considering the cost of the encoding can be important. When combining (reading, summing and writing) 120 shots from disk, which are then imaged and modeled, the cost of the encoding is almost as expensive as the rest of the procedure. This is because disk access can be very slow, especially when compared to an optimized GPU based propagator. To illustrate, a typical hard-disk used for this study could transfer to the CPU at around 200 Mb/s, then the CPU-GPU transfer speed saturates at 2 Gb/s, and the GPU (Fermi M2090) can operate at a bandwidth of 177 Gb/s.

Fortunately, there is a viable solution, which takes advantage of the multi-threading capability of CPUs. The encoding matrix α changes between each iteration, but each subsequent α is independent of previous choices, and of previous iterations. During GPU propagation, a separate compute thread can be used to prepare the encoded

data for the next iteration. Assuming encoding time is faster than the cost of the iteration, this will be hidden for all subsequent loops. For the first iteration waiting for encoding to complete is unavoidable.

RANDOMIZED SAMPLING

A different class of data-space reduction known as randomized sampling (Friedlander and Schmidt, 2012). This is often also referred to as stochastic dimensionality reduction, or as stochastic gradient methods (van Leeuwen et al. (2011); Aravkin et al. (2012a)). Conceptually, randomized sampling is very simple; between iterations a different subsection of the data is ignored. How much data is ignored, and how many iterations are required, will again be a function of survey geometry and shot density.

This concept has been analysed for both LSRTM (Dai et al. (2014); Leader et al. (2014)) and Full Waveform Inversion (FWI) (Aravkin et al. (2012b); van Leeuwen et al. (2011)). This discussion will briefly look at recreating similar results to these, and to then extend the concept to incorporate phase encoding.

How much data do we need?

Selecting the subset of the data is simply done by designing random function of zeroes and ones, with the number of ones corresponding to the percent of the input data to be used for the given iteration. This can be thought of as a separate encoding sequence of zeroes and ones. The same SEAM 120 shot 3D dataset was used, validating equivalent comparisons to the LSRTM and phase encoding results. Stochastic inversion was run using 75%, 50% and 25% of the data over a range of iterations and each was then normalized to the cost of one iteration using 100% of the data.

Figure 3.6 shows the same SEAM section after a single iteration for each subset. Predictably, using more data provides a better image for a single iteration. This is particularly noticeable in the depth slice. The same section after ten and 50 iterations are shown in Figure 3.7 and Figure 3.8. At high iteration counts, the difference

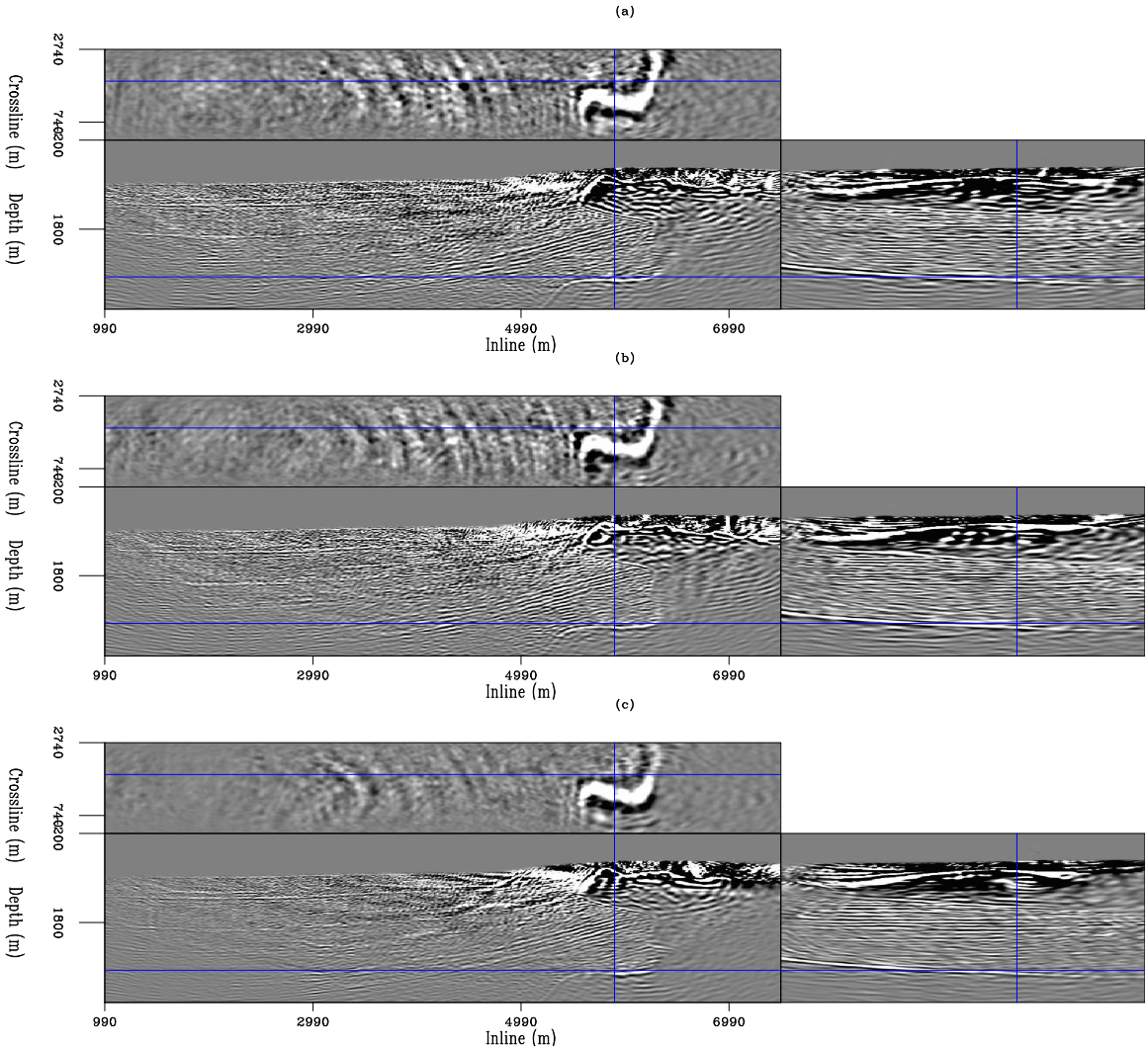

Figure 3.6: Images obtained after one iteration using 75%, 50% and 25% of the data, respectively. [**CR**] chap3/. randsamp1st

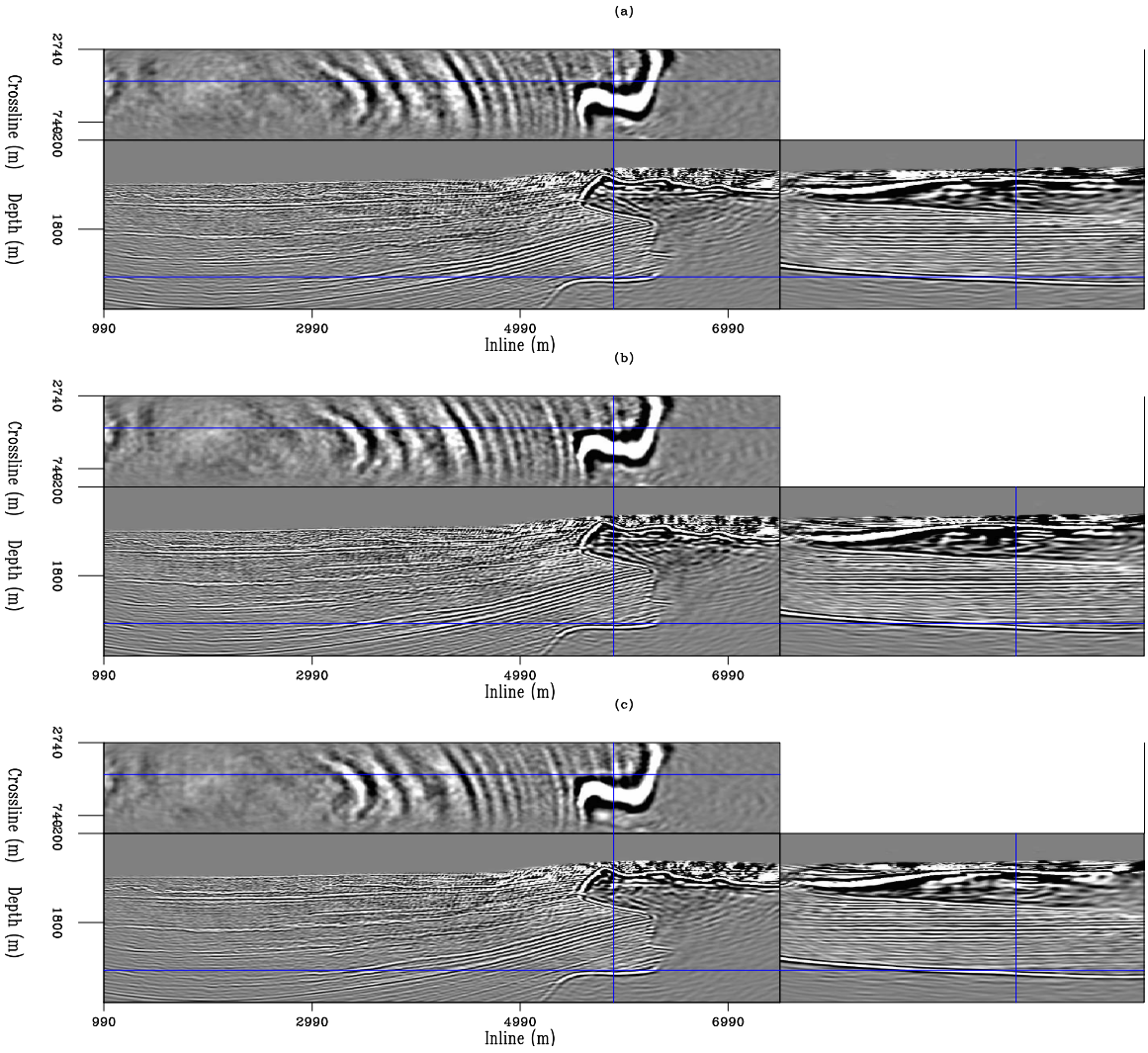

Figure 3.7: Images obtained after ten iterations using 75%, 50% and 25% of the data, respectively. [**CR**] chap3/. randsamp10th

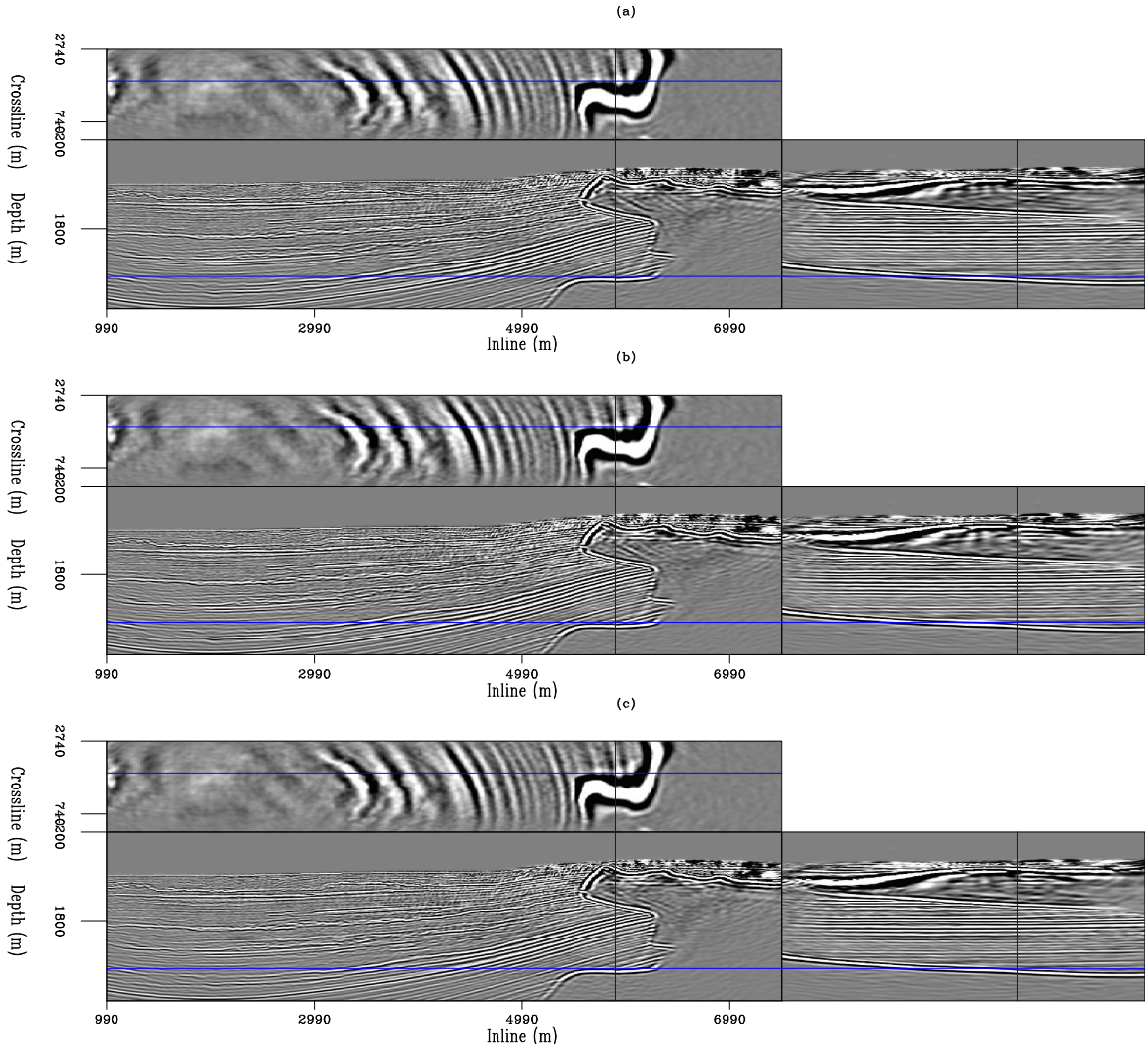

Figure 3.8: Images obtained after 50 iterations using 75%, 50% and 25% of the data, respectively. [**CR**] chap3/. randsamp50th

between these images becomes marginal, despite the widely favouring amount of data used per iteration. This shows that there will likely be a cost reduction for a given level of convergence.

Figure 3.9 shows some of these convergence curves, as a function of iteration number and quantity of data used. Similarly to phase-encoding, the 'observed' data change between iterations, creating a non-linear problem. This means that convergence properties are not smooth, and the residual can locally increase. It is noticeable that the trend for all subsets is a residual reduction. Also, if less data is used, then the less predictable the residual properties. This is because, at early iterations, sections of the model may not be well resolved, since only 25% of these data are being used. This unbalanced, noisy image, is then used for forward simulation.

It should also be noted that there is a concern with fair comparison. The raw initial residuals for these four sets of simulations were wildly different, and when plotted on the same scale, can be hard to interpret. For the sake of demonstration, these have all been scaled to percentages of data misfit. However, it should be noted that a given level of misfit for these varying data sizes are not directly comparable.

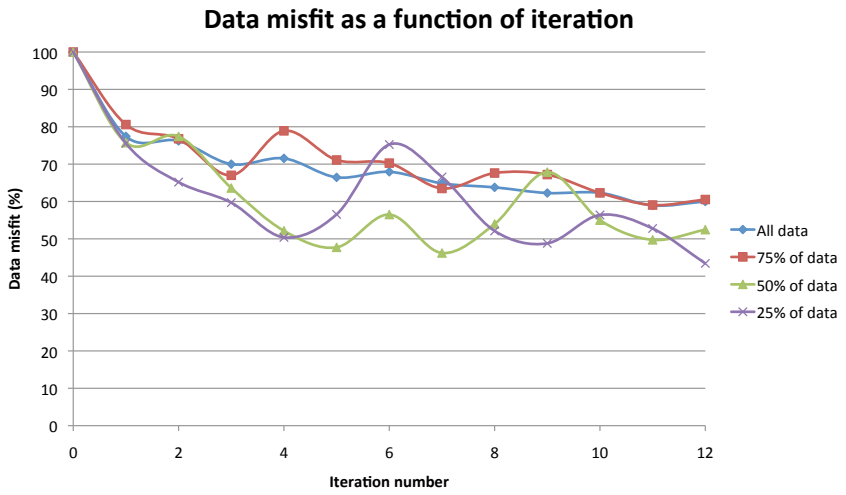

Figure 3.9: Plot of how the data misfit varies as a function of iteration number. [**NR**] chap3/. randsampiter

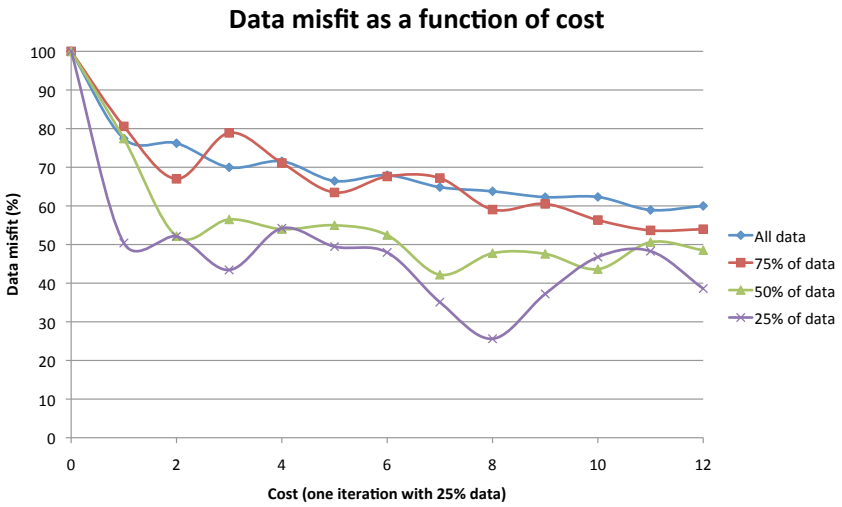

Figure 3.10: The same plot as in Figure 3.4, but as a function of cost. [**NR**] chap3/. randsampcost

Residual reduction as a function of relative cost can then be seen in Figure 3.10. It is clear, once again, that this stochastic dimensionality reduction has induced some local non-linearity, since microscopically the residual can often increase.

Making firm conclusions is difficult, since the problem is so dependent on source sampling. A dense source carpet would lend itself more easily to encouraging to a randomized sampling methodology than a sparse survey. The input dataset was relatively sparse, especially in the crossline, compared to many modern datasets. Since several iterations are being performed then in this case using 50% of the data gives the most favorable, and trustworthy, convergence. This ratio provided the right balance between source-sampling and inversion cost.

A more pronounced reduction would be seen for a denser survey, so these results show that even for a fairly sparse survey randomized sampling can be a powerful method to improve inversion feasibility.

PHASE ENCODING AND RANDOMIZED SAMPLING

When multiple super-shots are used for phase encoding, these two techniques can be combined. There will now be two loops of encoding, the inner loop being the phase encoding, and the outer loop being the randomized sampling.

Combining phase encoding with randomized sampling may have advantages since artifacts from phase encoding are incoherent between inner loop iterations, and the data reduction from the outer loop should save additional computation time.

CONCLUSIONS

Conventional linearized inversion is a very powerful method, however for large datasets, computational requirements can become unfeasible. By trying to reduce data movement, and the work done within each iteration, this feasibility can be improved.

Phase encoding addresses this by reducing the data-space dimensionality, without losing any information contained within these data. Shots are combined, imaged, and modeled concurrently. Crosstalk artifacts slow convergence as a function of iteration number, but by changing the encoding sequence between iterations, excellent convergence as a function of cost is observed relative to LSRTM (for a fixed receiver geometry.)

Optionally, a rolling subset of these data can be ignored within each iteration (randomized sampling.) This is quite different to the concept of phase encoding - some data is ignored, but no crosstalk artifacts are reduced. For dense source sampling an improvement in convergence as a function of cost is seen.

It is possible to augment these techniques. For certain survey geometries, this combination gives the most favourable data-space convergence, as a function of overall cost.

Chapter 4

Simultaneous shot separation

Typically, in seismic acquisition an active source is shot, and then a set waiting time must elapse before the next source point is acquired. This is to allow the energy to sufficiently dissipate, such that adjacent sources do not actively interfere. In simultaneous shooting, this restriction is relaxed (Beasley et al. (1998b); Stefani et al. (2007); Beasley (2008)). Despite this unvaoidable high energy overlap, many methods exist, or have been postulated, that can use these data very effectively. There are ways to design these surveys to minimise types of interference (Ikelle, 2001), to maximise acquisition potential (Abma and Ross, 2014), or to simply ignore the fact other sources are present (Krohn and Neelamani, 2008).

The concept of recording overlapping data has many terms in modern nomenclature, often meaning subtly different things. Allowing active sources to overlap may be called blending, simultaneous shooting, or continuous recording, amongst others.

Chapter 3 discussed phase-encoding in detail. Of course, there are stark similarities between simultaneous acquisition and phase-encoding. From hereon, the former will be referred to as 'primary' blending, and the latter as 'secondary' blending. This language is intuitive, since during phase-encoding the data provided were acquired conventionally, making the blending a secondary process.

For phase-encoding, how to blend and combine these data was a key consideration.

For field blended data, post-acquisition, the blending is fixed. There are encoding options available during acquisition - randomness of delays, waiting times, amplitudes, relative frequency contents etc, but once the user is given these data the blending can not be changed. This poses a variety of new problems for how to eventually image these simultaneously shot data, and how to optimally design a blended survey. This chapter will focus on designing a robust method for separating simultaneously shot data, under a variety of shooting patterns.

Initially, assuming the desired, final output is a clean, high-fidelity image, comparable to the image obtained from an unblended survey, then there are two clear options. One can attempt to directly image these data, or attempt to separate these overlapping data to approximate a conventionally acquired data-set, and image subsequently. The following sections will discuss the benefits and pitfalls from these two approaches.

IMAGING BLENDED DATA

Firstly, direct imaging of a simultaneously acquired data-set will be explored. As discussed in Chapter 2, either Reverse Time Migration (RTM), or linearised inversion can be used, the former being the first adjoint procedure of the latter.

Passive imaging

Naive migration of these data, as if they were unblended, has been increasingly referred to as 'passive' imaging of blended data (not be confused with the imaging of passive data). It follows from some simple algebra (and intuition) that the image will be plagued with cross-talk artifacts.

$$\sum_{i=1}^{n} \mathbf{L}_i \mathbf{m} \;=\; \mathbf{d} \tag{4.1}$$

If an operator, $\mathbf{L}$, describes all these data, as $\mathbf{Lm} = \mathbf{d}$, then $\mathbf{L}$ can be decomposed into individual shots, as shown in equation 4.1. Here, $\mathbf{m}$ describes the scattering potential field, $\mathbf{d}$ describes the entire data-set, and $\mathbf{L}_1$ relates the first shot, $\mathbf{d}_1$ to $\mathbf{m}$ etc., up to shot n.

Now, consider the case where $\mathbf{d}_1$ and $\mathbf{d}_2$ overlap, such that $\widetilde{\mathbf{d}} = \mathbf{d}_1 + \mathbf{d}_2$. Conventional migration applies $\mathbf{L}'$ to these data, where $\mathbf{L}' = \mathbf{L}'_1 + \mathbf{L}'_2$, and sums the result to form an estimate of $\mathbf{m}$. The result of this operation is formulated in equation 4.5.

$$\begin{aligned}
\mathbf{m} &\approx \mathbf{L}'\widetilde{\mathbf{d}} && (4.2)\\
&\approx \mathbf{L}'_1\widetilde{\mathbf{d}} + \mathbf{L}'_2\widetilde{\mathbf{d}} && (4.3)\\
&\approx \mathbf{L}'_1\mathbf{d}_1 + (\mathbf{L}'_1\mathbf{d}_2 + \mathbf{L}'_2\mathbf{d}_1+)\mathbf{L}'_2\mathbf{d}_2 && (4.4)\\
&\approx \mathbf{m}_0 + \widetilde{\mathbf{m}}_0 && (4.5)
\end{aligned}$$

The imaging artifacts come from the operator that images $\mathbf{d}_1$ being applied to $\mathbf{d}_2$, and vice-versa. It is clear why this interference is known as crosstalk, since it comes from these cross-terms.

There are circumstances under which direct migration of these data results in an acceptable image. If shots are well separated by distance (Distance Separated Simultaneous Source, or DS^3, Bouska et al. (2009)), then this operator overlap is small, and crosstalk artifacts migrate outside of the domain of interest (or at least far from the imaging target). Furthermore, if the blending power is low (a relative measure of interference quantity), and the shooting dense, then the power of the stack will often mitigate these crosstalk artifacts. Since, whilst they are high amplitude on a shot-by-shot basis, they are incoherent between shots. The image may be unreliable for amplitude analysis, but for structural information, or for an initial model during inversion, it will be adequate. This is contingent on random time delays between sources, constant delays may not result in these artifacts stacking out as efficiently.

Whilst these two situations may result in adequate structural images, two problems remain. A better solution is needed for trustworthy amplitudes, and there is an implicit background velocity model assumption. For direct imaging, especially linearised inversion, a representative velocity model is needed. This point will be elaborated upon during the rest of this chapter.

Imaging through inversion

Passive imaging of blended data can be improved by simultaneous inversion. Options include direct inversion for the scattering model (Berkhout (2008); Tang et al. (2009)), target oriented simultaneous inversion (Verschuur et al., 2009), inversion for separate time-lapse vintages (Ayeni et al., 2009), amongst others. Conceptually, this is similar to phase encoding, where the encoding is fixed, and α is a vector filled with ones.

From the discussion in Chapter 3, one can conjecture that convergence for this encoding sequence would be slow. Meaningful results can be recovered, however, by virtue of the fact the model space is still vastly smaller than the data space.

However, there is a more fundamental limitation with inverting blended data directly than slow convergence. Such a methodology would require accurate velocity control, which is crucial for linearised inversion and it's subclasses. If the velocity is not well constrained then the inverted scattering potential will be imaged incorrectly, and information will be lost. As an acquisition solution a stringent velocity requirement is unacceptable. Consequently, using direct inversion to image blended data is an interesting academic problem, but ultimately other solutions must be sought. This leads to the problem of simultaneous data separation.

SEPARATING BLENDED DATA

The most flexible, and velocity independent, option for processing overlapping data is to separate shots to individual records. Once this has been achieved a conventional processing flow can be followed. A methodology that is not strongly dependent on

velocity or Earth model knowledge is desirable, since for exploration surveying a strong control on subsurface geology is an unreasonable expectation.

For shot separation, several approaches are possible. The concept of image space inversion will be the focus of this thesis discussion; other, existing, options will be briefly discussed and contrasted first.

Data space filtering

A cursory glance at simultaneous shot records may give an observer the notion that simple data space filtering could be used. For relatively flat geology, and with well distance-separated simultaneous sources, this is often the case. Successful data space filtering methods rely on either conflicting dips, or using the apex of events to ascertain the shot of origin. However, once sources are proximate in both time and space, correctly identifying the origin of certain events in a shot record becomes difficult, and the dips of events can become more aligned. If the human eye is not able to distinguish these events, assuming there is limited prior model knowledge, then an algorithm will similarly fail. This is exacerbated for certain geologies, for example steeply dipping salt bodies. Reflections from these will appear at large offsets and late times, potentially causing them to become confused with other sources.

As an example, a group of randomly delayed shots are shown in Figure 4.1. In this figure it is largely possible to distinguish which shots are related with which reflection events. This can be contrasted with Figure 4.2, which shows two shots simulated over the same model, adjacent to a steep salt body.

The concept of separating well distance-separated shots is almost moot. Filtering antithetic events is trivial, and often simply migrating these data as if there was no overlap provides reasonable result.

The majority of existing and effective techniques rely on using a domain transform, or series of transforms, which can isolate coherent parts of the signal. These all originate from the concept of randomised source intervals and receiver gathers (Beasley

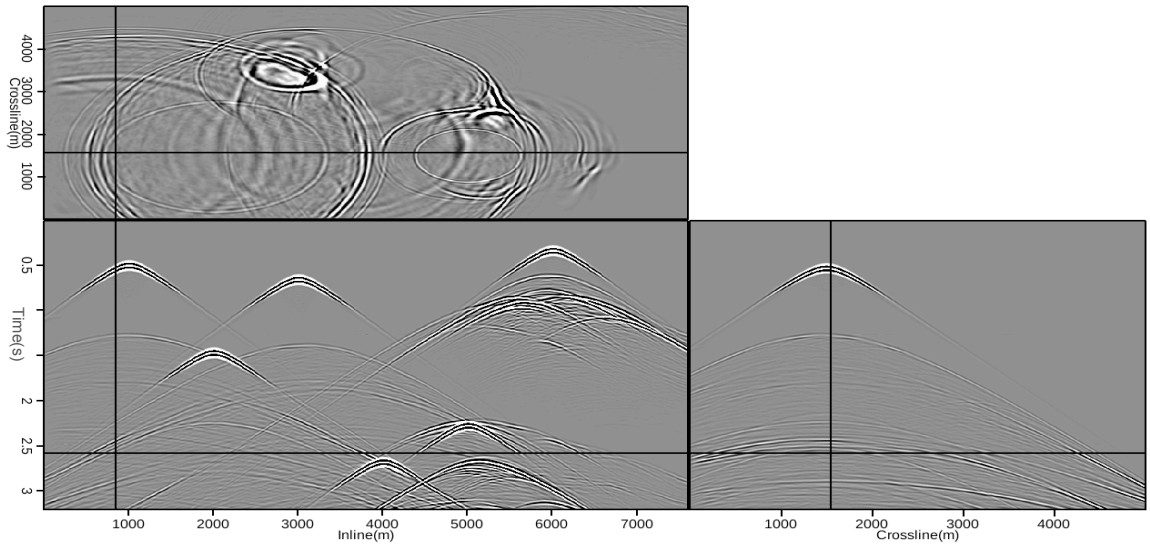

Figure 4.1: Five randomly delayed shots, while the record is noisy it is possible to distinguish which shot caused which event. [**CR**] chap4/. seamranddelays

et al., 1998b). The pinciple of seismic reciprocity concludes that, for a given shot record, the same data would be recorded if all sources and receivers were swapped. In the situation where overlapping shot points are randomized, in time and space, then transforming to the common receiver domain will scramble much of the overlapping energy, resulting in what could be considered as a shot gather plagued with random noise. A variety of random noise reduction technqiues can now be invoked.

This can be further illustrated by looking at 2D gathers, for example in Figure 4.3 and Figure 4.4. Here the right hand panel is a corollary to the receiver domain, the left to the shot domain. The difference in coherency is extremely evident when comparing the top and bottom panels.

The bottom panel in Figure 4.4 highlights another potential pitfall, if the delays are constant then there are no coherency differences between these domains. As a result, any attempt to treat overlapping shots in the receiver domain as random noise will fail, and primary information will be lost.

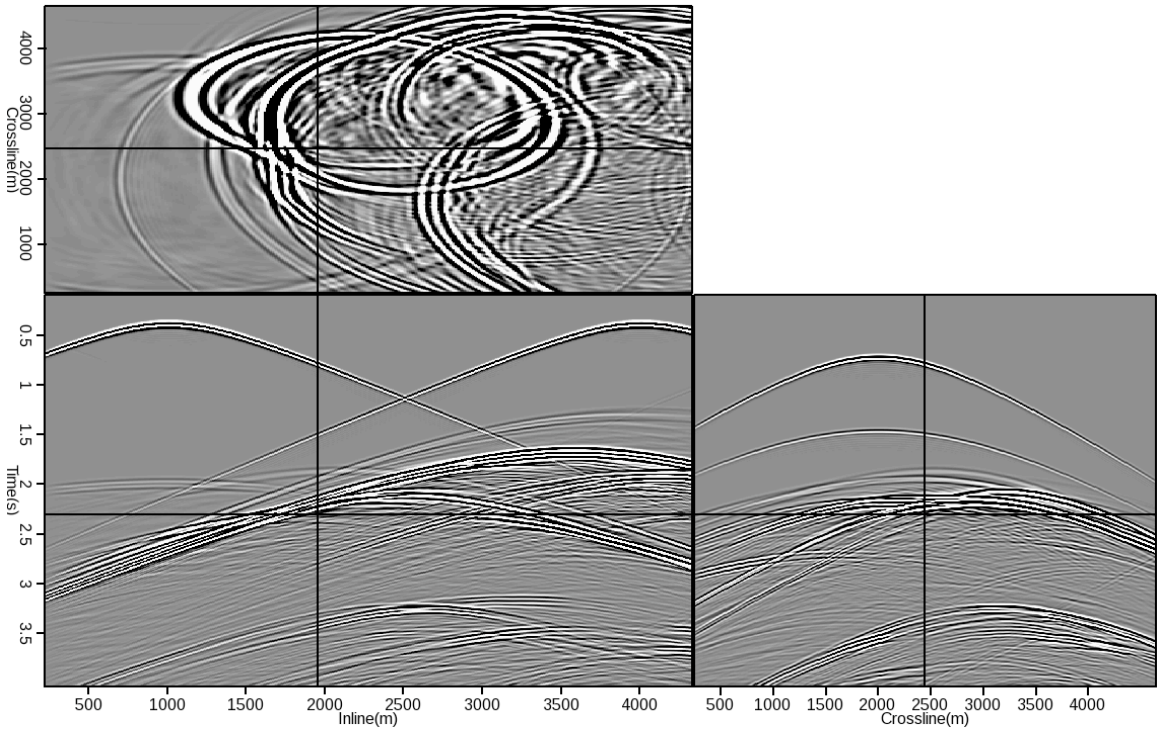

Figure 4.2: Two shots, simulated over the same model as Figure 4.1. At late times, it becomes impossible to ascertain which shots given events originated from. [**CR**] chap4/. couldweseparate

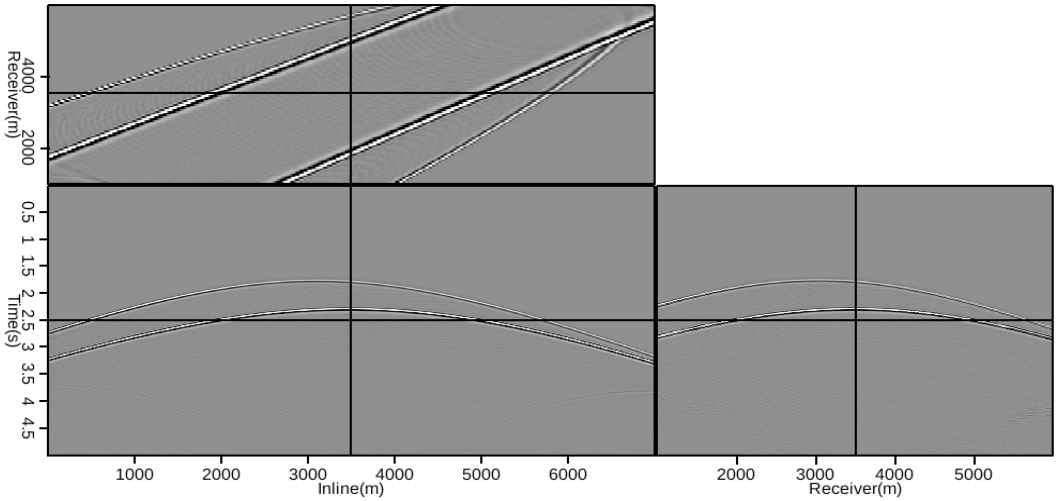

Figure 4.3: A simple dataset, from a two layer model. The left side panel is a shot gather example, the right side panel is a receiver gather. [**CR**] chap4/. simpledata

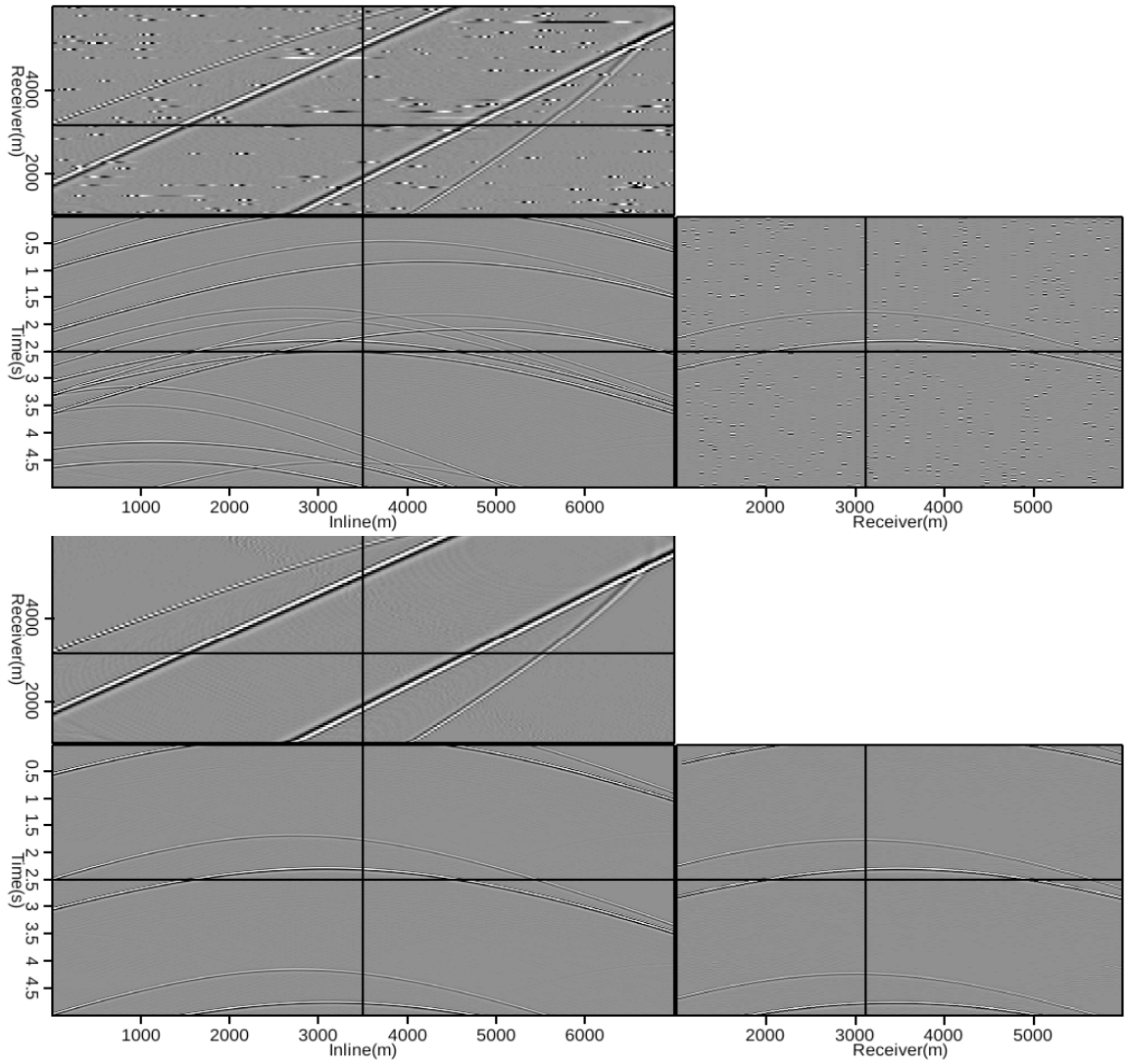

Figure 4.4: The same dataset as in Figure 4.3, after random blending (top) and linear blending (bottom). [**CR**] chap4/. receivergathers

Data space inversion

For randomised source timings, inverse methods based in the data space can be very effective in the field of simultaneous source separation; most of these use a variety of domain transformation, as mentioned above. Abma et al. (2010) showed that using a Projection Onto Convex Sets (POCS) methodology, in the common receiver domain, results in a high degree of separation. Whilst effective, this technique involves many multi-dimensional Fourier transforms (Abma, 2012), so for certain geometries and blending schemes separation as a function of compute time may be slow.

Other receiver-gather projections can also be used. Both median filtering (Huo et al., 2009) and sparse radon transforms (Akerberg et al., 2008) are effective for many survey types and can be powerful tools for noise reduction. Additionally, threshholded interference reduction (Doulgeris et al., 2010) and interference estimation (Mahdad et al., 2012) show promising results for data blended in 2D. Approaches borrowed from multiple prediction literature have been shown to have varying results, but often prediction-and-subtraction methods have yielded excellent separation (Spitz et al., 2008).

Ikelle (2010) looked at a variety of advanced blending patterns and analysed how to invert for the transform that will link the source and receiver domains. Other successful methods use a compressive sensing approach (Herrmann et al., 2009), where many wavelet transforms can be used to separate these data.

All these existing methods strongly rely on random source delays, and any repetition or predictability in the source timings can induce surprisingly detrimental artifacts. Abma et al. (2012) showed a number of interesting observations from field blending tests. Natural variation in the recharge time of these airguns can often provide a sufficient level of shot randomness. However, as survey size increases, so does the risk of underlying shooting predictability occurring in certain shot sequences. This is counter-intuitive in the field of seismic imaging, since generally an increase in survey size results in more redundancy and higher fidelity. It is possible to mitigate

this potential repetition by inducing more unpredictability, such as by adding an extra time delay. This can be determined from a random number table, and induced once the airgun is recharged.

Theoretically, these inverse methods can handle many interfering shots. In practice, for marine acquisition, no more than three or four interfering sources are present at a given instant. This is a function of physical space, and availability of source boats. On land many vibroseis trucks have been used at once for a high degree of parallel acquisition (Jack et al. (2008); Pecholcs et al. (2010)). It should also be noted that land acquisition provides a few other advantages for separation; different sweeps can be encoded to encourage orthogonalit and source positions are easier to randomise.

For these reasons, simultaneous land acquisition has been used extensively, and shown to give impressive results. There has been a marked increase in simultaneous marine acquisition, but some issues remain. For these reasons subsequent sections will be limited to the discussion of marine surveys, and will focus on a blending power of around three.

The image space

It is desirable to choose a domain, or inversion technique, which is less restrictive of blending pattern. In particular, developing a scheme which is robust while separating data acquired with constant or pseudo-constant delays is sought, since these aforementioned data-space methods fail under these conditions. By projecting the problem into the extended image space, these previous shortcomings can be circumvented. Although RTM is used as the transform operator, the use of an extended domain (either in offset or angle) loosens the restriction of having an accurate velocity model, which was alluded to in Chapter 2. No information is lost this way, and correct data reconstruction is achievable.

Furthermore, a technique that uses the image space can be easily augmented with a velocity update scheme - most of the necessary computation will already be done.

This strengthens the potential of such an approach, as an improved velocity model will allow for cheaper, more accurate shot separation, during the next set of iterations. This will be elaborated upon subsequently.

Through image space projection, 'primary' shots of interest can be focused on (migrated, essentially), while overlapping energy, from 'secondary' shots, will be dispersed. This allows separation criteria to focus on both move-out differences and focusing contrasts. This is simply done by combing the input data into discrete shot records, according to a suggested recording time length and a set of shot timings. This results in a data-set which is the size of the would-be conventional data, but plagued with overlapping, coherent events.

Using the extended image space to remove coherent noise sources has been previously analysed, albeit under other objectives. Alvarez (2007) looked at using apex-shifted parabolic-Radon transforms to successfully remove surface related multiples. Wong (2014) looked at using F-K transforms from the extended image space to successfully remove imaging noise. However, in these cases the final imaging procedure had been completed, and this step was to remove some coherent artifacts before stacking to the zero-offset image.

The implicit goal herein is a separated dataset. As a result, transforming from the extended image space back to the data space is necessary - this is sometimes termed demigration (Tang and Biondi, 2011). Using demigration for multiple removal has been effective in several studies (Sava and Guitton (2005); Zhang and Duan (2013)), and this is a similar problem to that of source separation. For amplitude consistent recovery, one pass of extended Born modeling (demigration) is insufficient, and inverse demigration is necessary (Chauris and Benjemaa, 2010). The process of demigration is equivalent to Born modeling, so for clarity the term inverse Born modeling will be used to describe linearized forward modeling as an inverse process, rather than inverse demigration.

Such inverse Born modeling methods have been shown to give full data recovery (Sava and Vasconcelos, 2011), and to remove multiples (Weibull and Arntsen, 2014),

although it has been an only lightly explored subject. To formally introduce this concept Born modeling must be revisited, and the extended image space will be introduced in full.

The extended image space and Born modeling

$$I(x,y,z) = \sum_{i}^{nshots} \sum_{t} P_s(x,y,z,t;\mathbf{s}_i) P_r(x,y,z,t;\mathbf{s}_i), \tag{4.6}$$

$$I(x,y,z,x_h,y_h) = \sum_{i}^{nshots} \sum_{t} P_s(x+x_h, y+y_h, z, t;\mathbf{s}_i) * P_r(x-x_h, y-y_h, z, t;\mathbf{s}_i). \tag{4.7}$$

If equation 4.6 describes Claerbout's zero-offset imaging condition, then an example of extended imaging can be described by equation 4.7. Here, $I(x,y,z)$ is the image in space, P_s is the source wavefield and P_r is the receiver wavefield; s_i represents the current shot of interest. If lag coordinates (known as subsurface offsets) in x and y are introduced (x_h and y_h), a 5D image can be created. It is possible to have lags in both t and z to create a 7D image, or any combination thereof.

If the correct velocity is used for extended migration, then an image sharply focused at zero-subsurface-offset is created. All the necessary kinematic and amplitude information for data reconstruction is contained in this zero-subsurface-offset image panel, an example can be seen in Figure 4.5. Artifacts present are a result of limited acquisition and data truncation, and will not hinder reconstruction. Next, Figure 4.6 shows the same data, but migrated using a constant velocity model (incorrect up to 20%). This zero-offset panel is now not sharply focused, and coherency over a range of subsurface offsets is observable. In this case much of the necessary kinematic and amplitude information, critical for accurate extended Born modeling, is spread over these offsets. Consequently, these must also be back-projected for accurate data

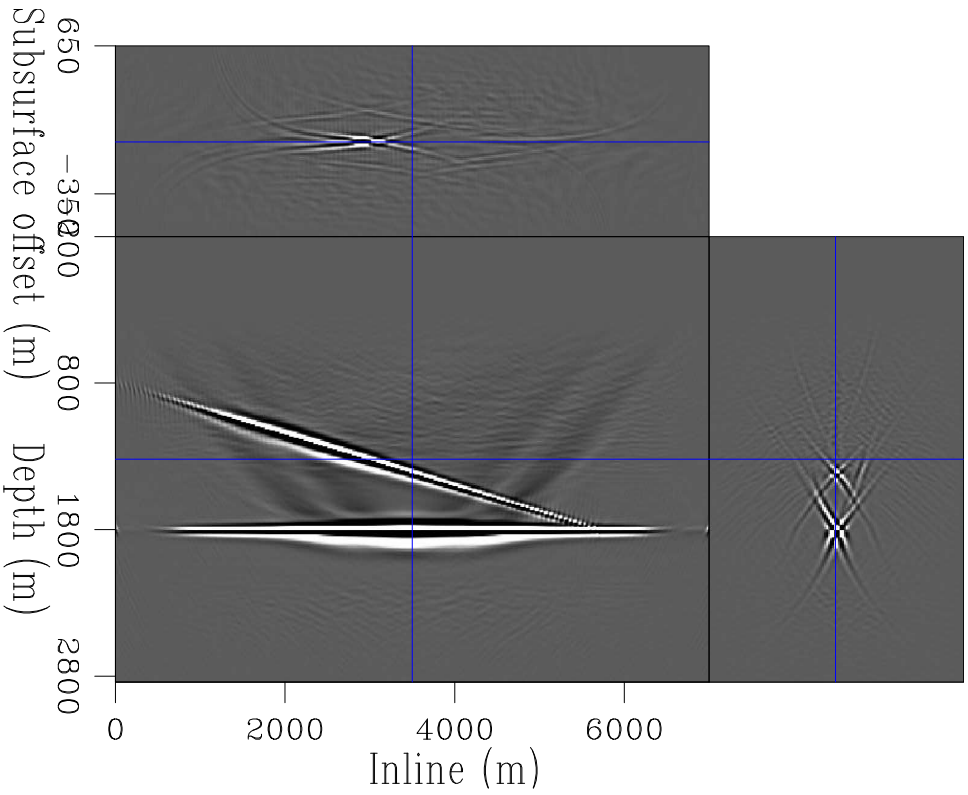

Figure 4.5: The unblended simple dataset, as seen in Figure 4.3, migrated into the extended domain with the correct velocity. [**CR**] chap4/. extsimpimgcv

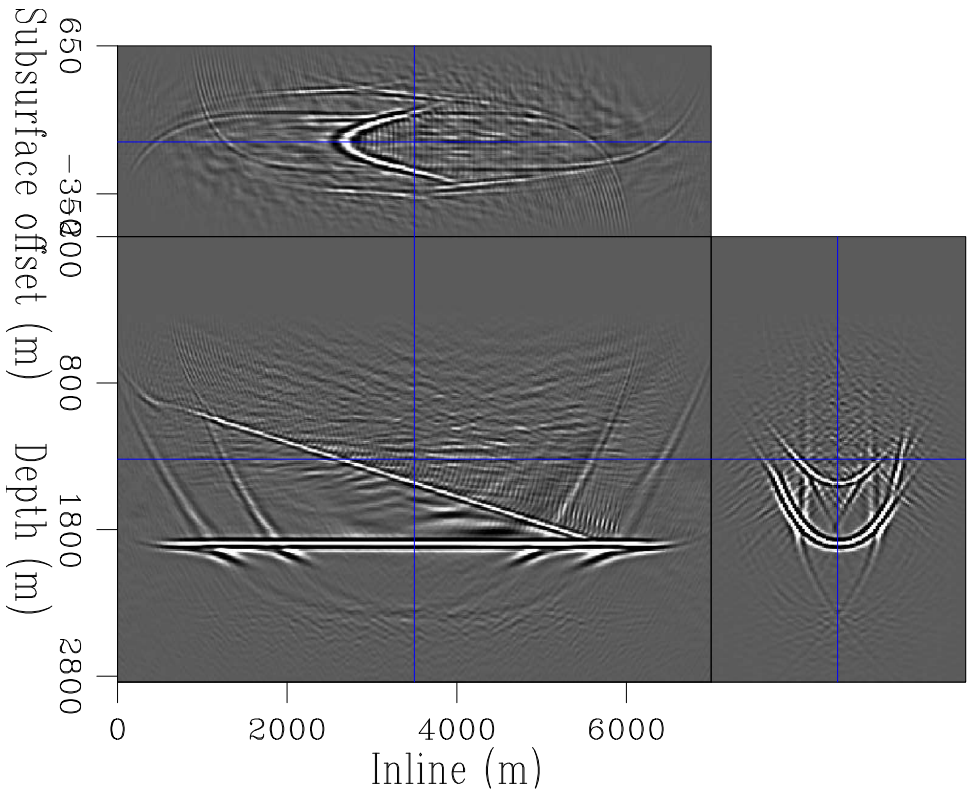

Figure 4.6: The unblended simple dataset, as seen in Figure 4.3, migrated into the extended domain with using a constant velocity model. [**CR**] chap4/. extsimpimgincv

recovery.

It is possible to apply an additional transform from subsurface offset to the angle domain. By measuring event curvature as a function of this opening angle, velocity model updates can be estimated. It is by this approach that Wave-Equation Migration Velocity Analysis (WEMVA) works (Sava et al., 2003). For many algorithms the angle domain is used instead of the subsurface offset domain, since this parameterisation is directly relatable to surface geometry and velocity. The concept of suburface offset itself is somewhat unphysical, but during extended imaging it provides the necessary degrees of freedom to preserve input wavefield information.

In addition to this simple, two-layer dataset, two more complex and geologically representative models will be used. These are a slightly modified version of the reference Marmousi model (Versteeg, 1994) (padded with a water layer), and a windowed subsection of the latest SEAM model (Fehler and Larner, 2008). Initially, the Marmousi model will be used to investigate if this added complexity poses additional difficulty for inverse Born modeling, and how these results vary under different blending experiments. The velocity models used, and the reference unblended dataset, are shown in Figure 4.7 and Figure 4.8. For this dataset, 60 shots were simulated with a spacing of 100 m, receivers were all along the top of the model.

Random delays

Random delays between source times and positions provides the most options for data separation. Figure 4.2 already showed the simple two-layer dataset, with both random and constant delays simulated. The implications of these scenarios will now we analysed in more detail.

The left-side panels (of Figure 4.2) show examples of shot gathers, there is a significant amount of highly coherent noise throughout these records. The right-side panel shows example receive gathers (since the recorders were fixed.) Noticeably, for random delays the overlapping noise takes on a highly different characteristic - the data of interest (primary events) are coherent, and all of the contamination has now

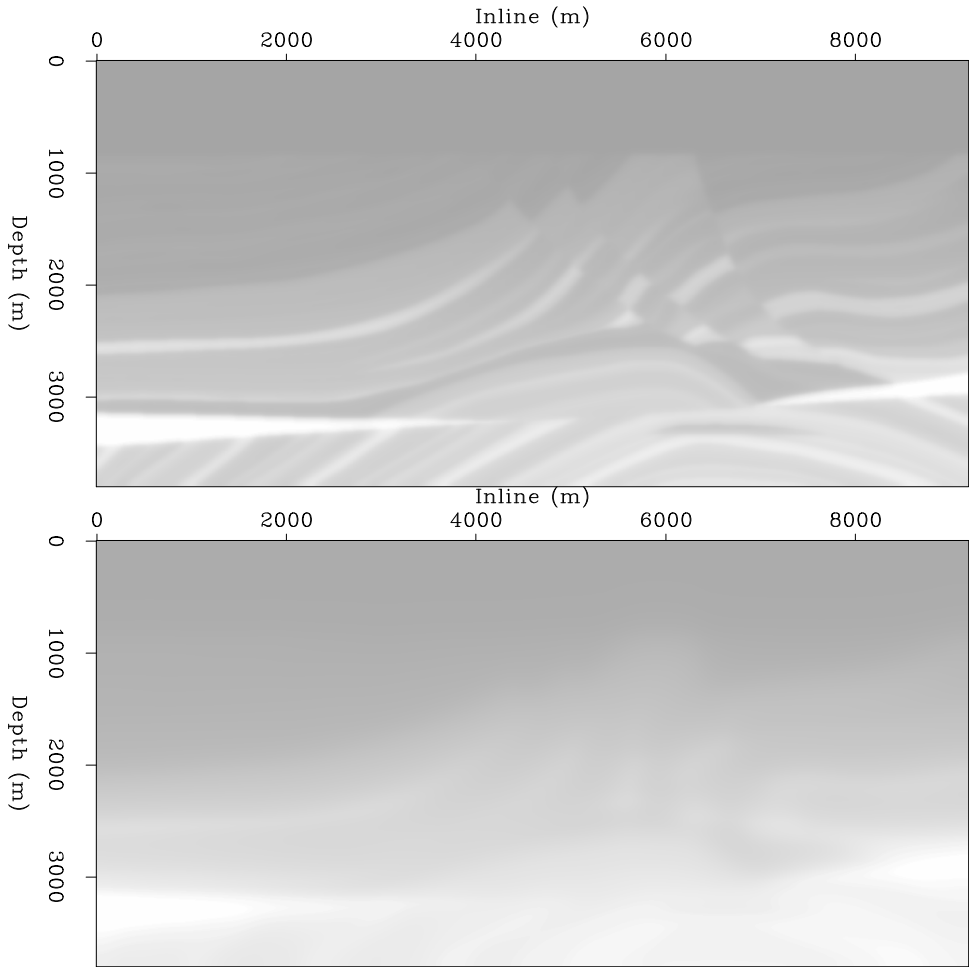

Figure 4.7: Accurate and smoothed representations of the Marmousi velocity model. [**ER**] chap4/. marmvels

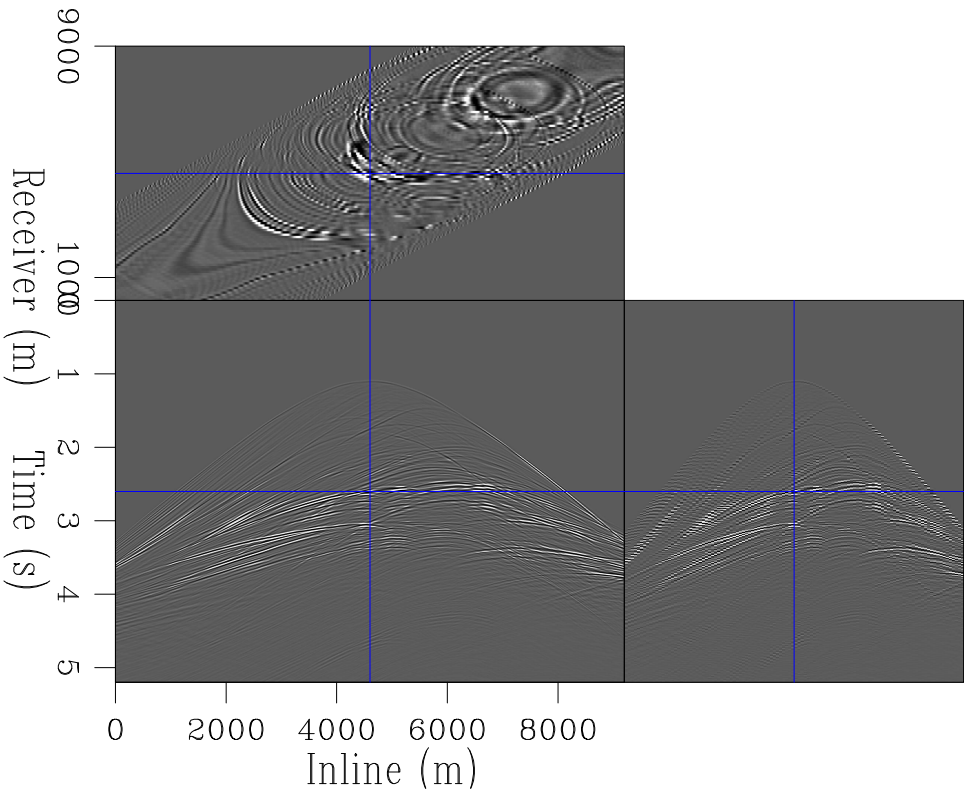

Figure 4.8: Data acquired over the Marmousi model. [**CR**] chap4/. marmdata

been incoherently scrambled.

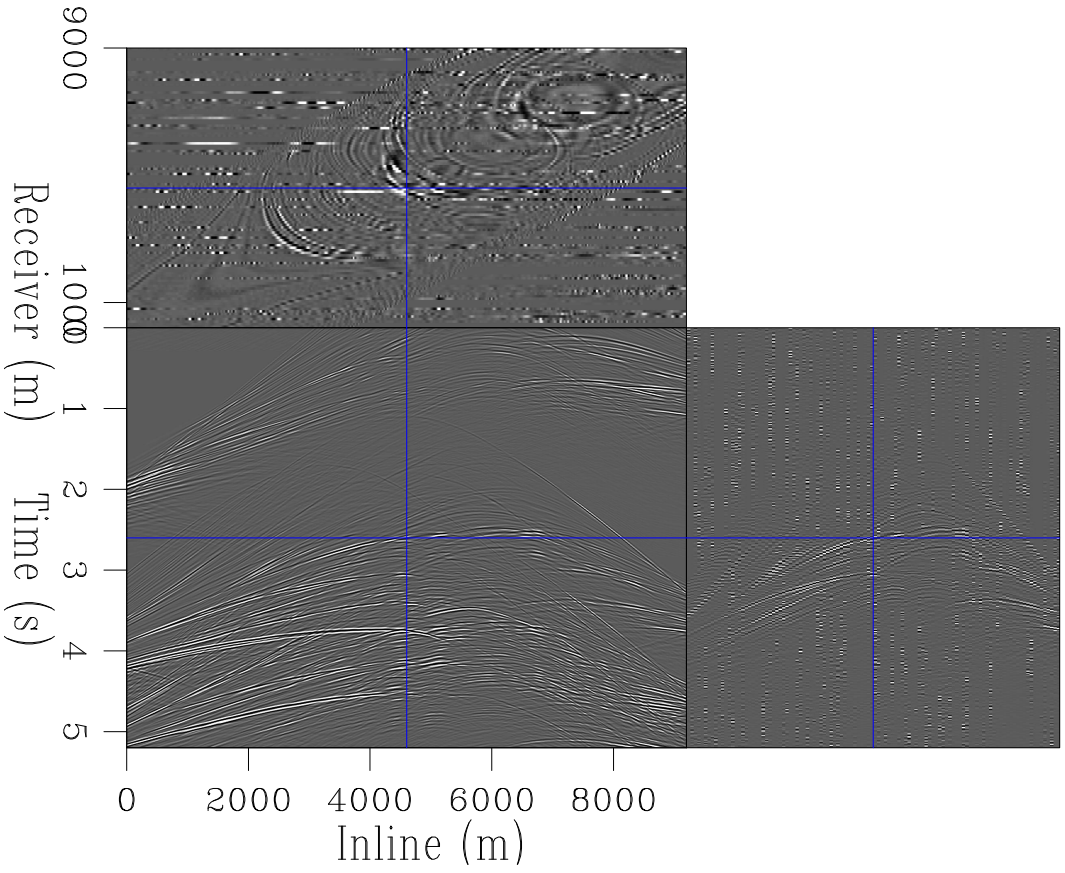

Figure 4.9: Data acquired over the Marmousi model using a random delay function. [**CR**] chap4/. marmrndblnd

Inducing small delays between sources was an early style of simultaneous acquisition, known as dithering (Moore et al., 2008). Scrambling this noise was done through a simple, and very cheap, domain transformation, typically to an F-K space. The noise is spread over a large range of spatial and temporal frequencies, whilst the primary information is more tightly contained. This can be exploited in a number of ways - filtering, threshholding, inversion etc.

Figure 4.10 then show these randomly delayed data after imaging, migrated with the correct velocity model. The majority of this overlapping energy has stacked out, but there is a significant amount of high-wavenumber noise remaining in the image.

This makes this simple imaging acceptable for structural / model building work flows, but will be insufficient if any amplitude work is desired.

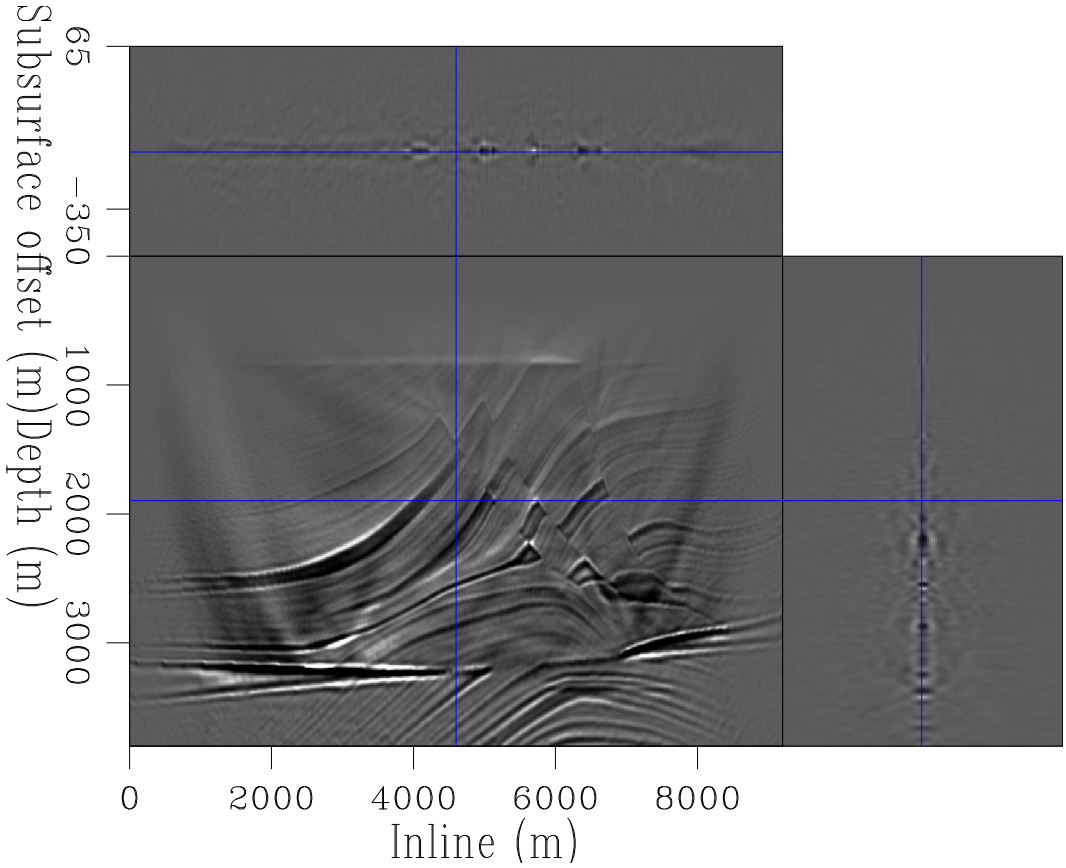

Figure 4.10: The extended image from migrating Figure 4.9 (data acquired using a random blending function) [**CR**] chap4/. marmrndblndim

Linear delays

A constant shift between source locations and timings will be denoted as 'linearly' delayed data, and this can pose some interesting, additional problems.

Figure 4.11 shows these Marmousi data, but this time the shooting pattern features a constant delay. Thus, there is no incoherency to exploit in the common receiver domain. This is further illustrated by looking in the F-K domain, the signal

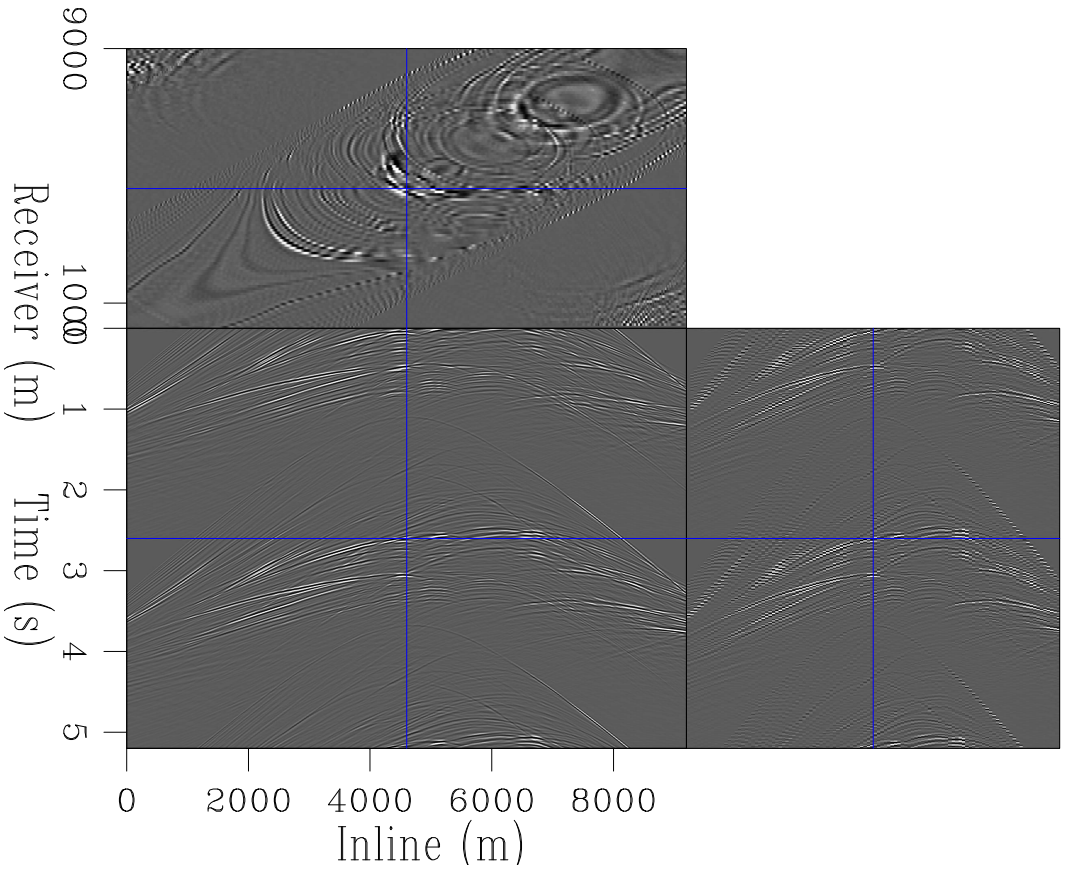

Figure 4.11: Data acquired over the Marmousi model using a linear delay function. [**CR**] chap4/. marmlinblnd

and the noise remain tangled, and techniques like filtering and threshholding will fail at removing the noise, and may remove primary information.

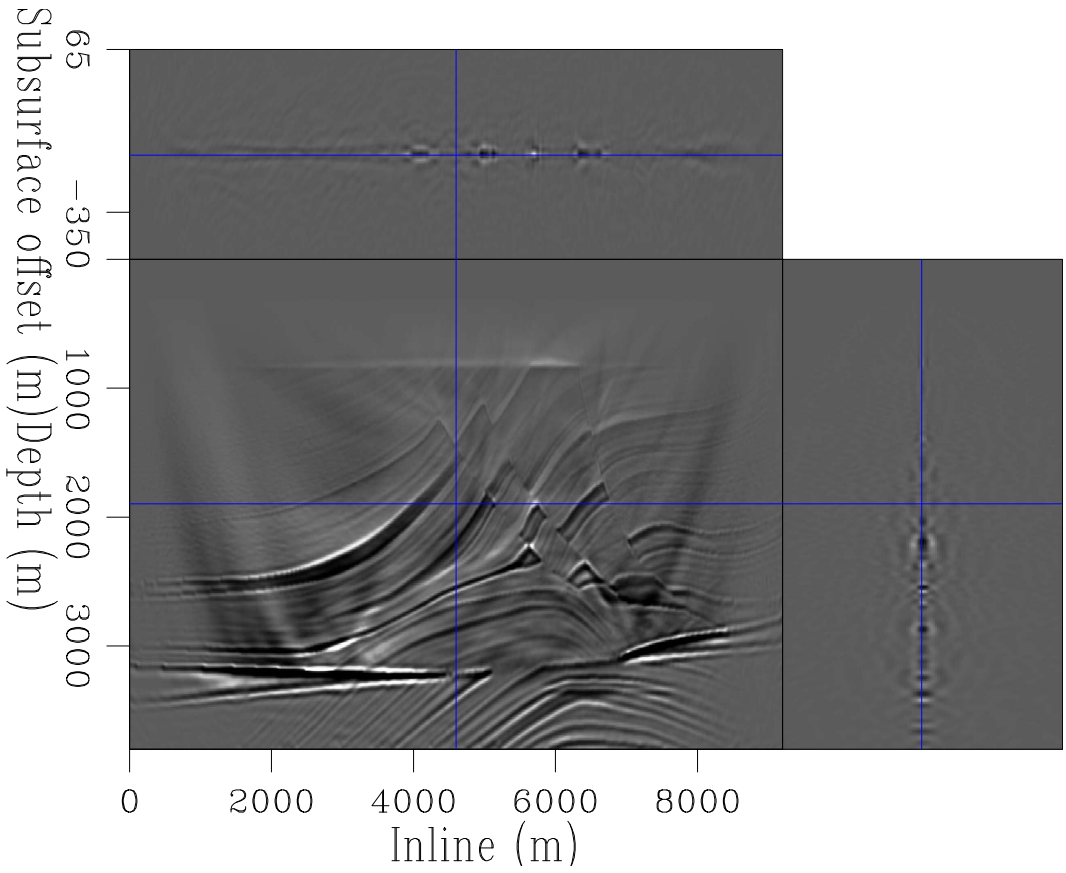

Figure 4.12: The extended image from migrating Figure 4.11. [**CR**] chap4/. marmlinblndim

Imaging these data, with the correct velocity, gives the result in Figure 4.12. Somewhat surprisingly, the image is not plagued with high-amplitude artifacts. The reason for this is a little more subtle - because the blending style was not very aggressive, a lot of the noise either does not correlated into the image, or migrates outside of the domain of interest. These is still some noise, relative to the conventional image, but this is less damaging than in the random case. Additionally, these artifacts are comparably high-amplitude if single-shot images are observed, however these artifacts are still incoherent between images. Thus, stacking removes much of the noise which

was migrated into the target area.

Pseudo-linear delays

The most realistic marine acquisition scenario is often that of pseudo-linear delays. These are close to constant shifts, but with a jitter of around 5% between times.

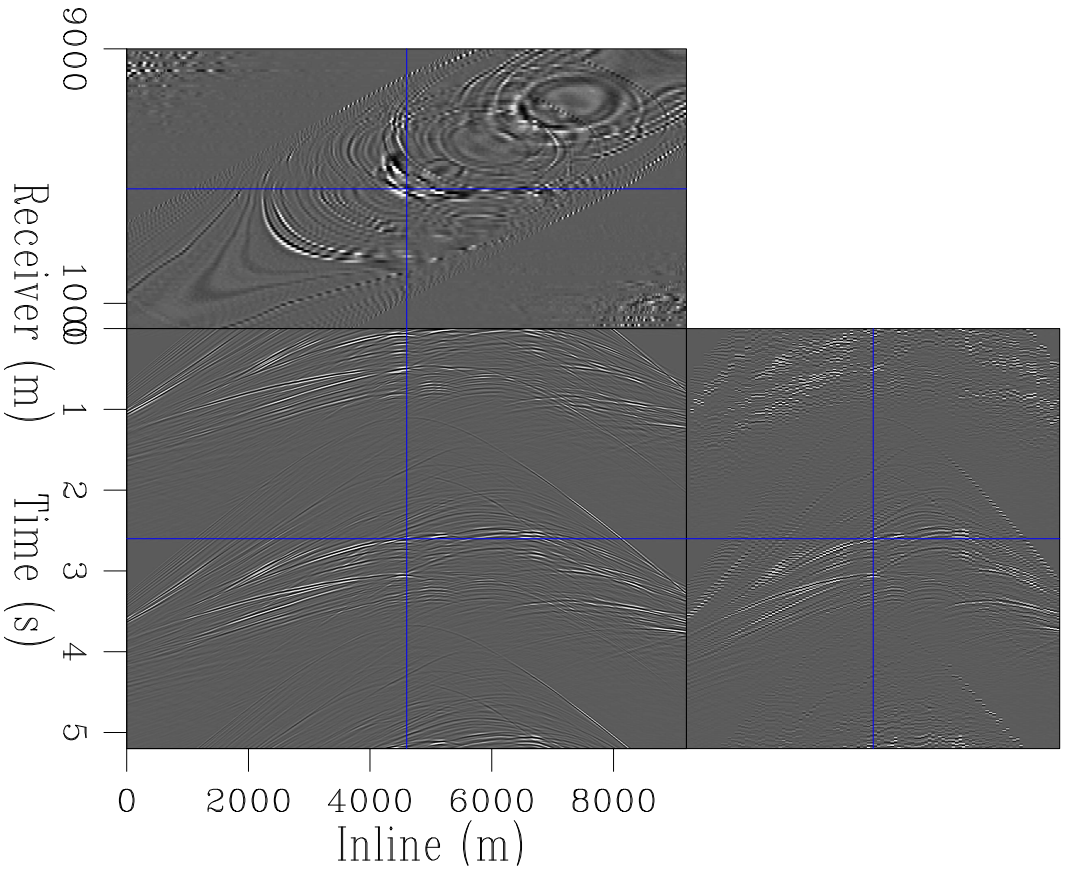

Figure 4.13: Data acquired over the Marmousi model using a pseudo-linear delay function. [**CR**] chap4/. marmplinblnd

This small amount of randomness creates a high level of incoherency between domains; migrating these data gives similar results to the linear delays, and again, these are cleaner than for random delays. Largely, this is due to the temporal separation of the signal and noise. The slight randomness then provides the additional incoherency

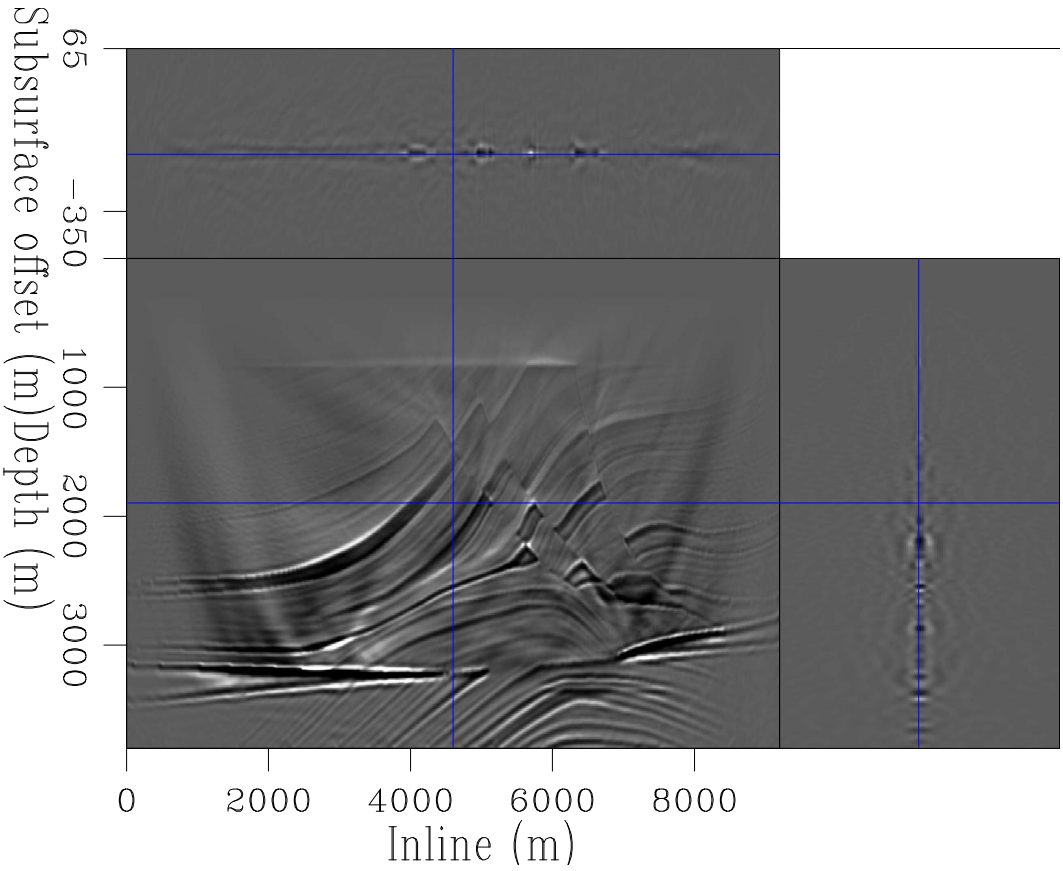

Figure 4.14: The extended image from migrating Figure 4.13 (data acquired using pseudo-linear delays). [**CR**] chap4/. marmplinblndim

needed to largely stack out the noise.

Angle domain inversion

From studying the two-layer model subsurface offset panels, particularly under linear blending conditions, it appears that blended energy could be isolated as a function of curvature, and hence removed. If it is possible to remove contaminated energy from these panels, then the images could be demigrated, and data separation achieved. To test this theory, this two-layer dataset, with linear blending, will be used. Figure 4.15 shows the correct velocity image, and an image constructed from a velocity which was scaled to be too slow. In both of the subsurface offset panels, events from the data of interest ('primary' events) and events from the blended energy ('secondary' energy) are both easily identifiable. However, for the incorrect model the events now all have significant curvature, and the focusing contrasts are less pronounced.

A second domain transform is necessary, such that events can be isolated as a function of curvature. A parabolic Radon transfrom was selected, since for a flat reflector the curvature of these events should approximate to a parabola. However, applying a parabolic Radon transform to the top panel will be problematic, since some events have been focused to a point - this will cause this information to become spread out in the Radon domain. Instead, another transfrom must be used before moving to the Radon domain, and this is a transform to the angle domain. Measuring the curvature of these events in the angle domain is how WEMVA updates are constructed.

The angle domain representation is shown in Figure 4.16. The previously well-focused events now appear flat, and the curvature of other events have become more extreme. It is now possible to transfrom to the parabolic Radon domain, and isolate events by this curvature.

However, a parabola is not an ideal representation of these events, and nor is the Radon transform used very exact (due to aperture restrictions). These limitations are manifested in Figure 4.17; whilst some curvature separation has been achieved,

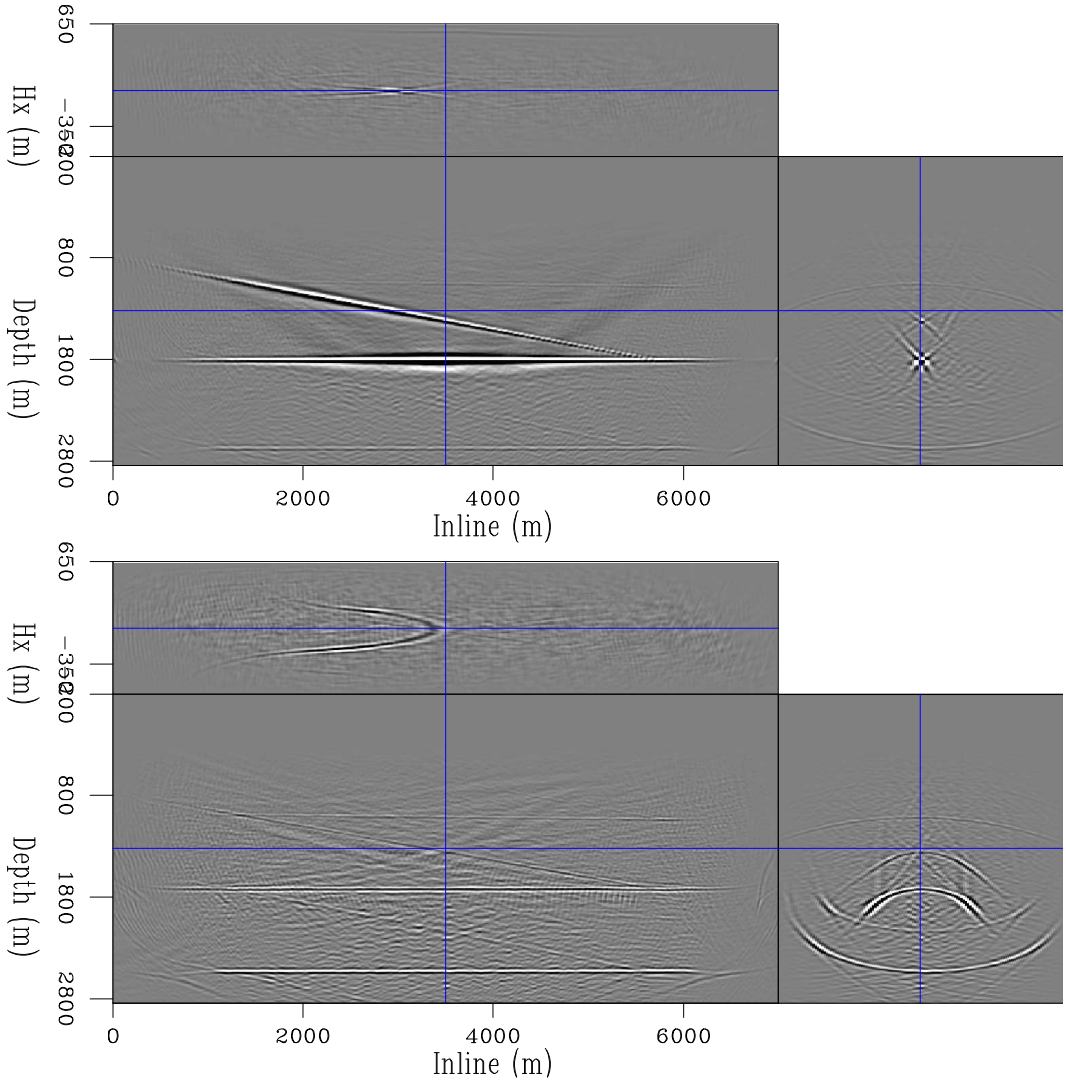

Figure 4.15: Linearly blended data migrated into the subsurface offset domain using the correct velocity model (top) and a model 10% too slow (bottom.) [**CR**] chap4/. offimgs

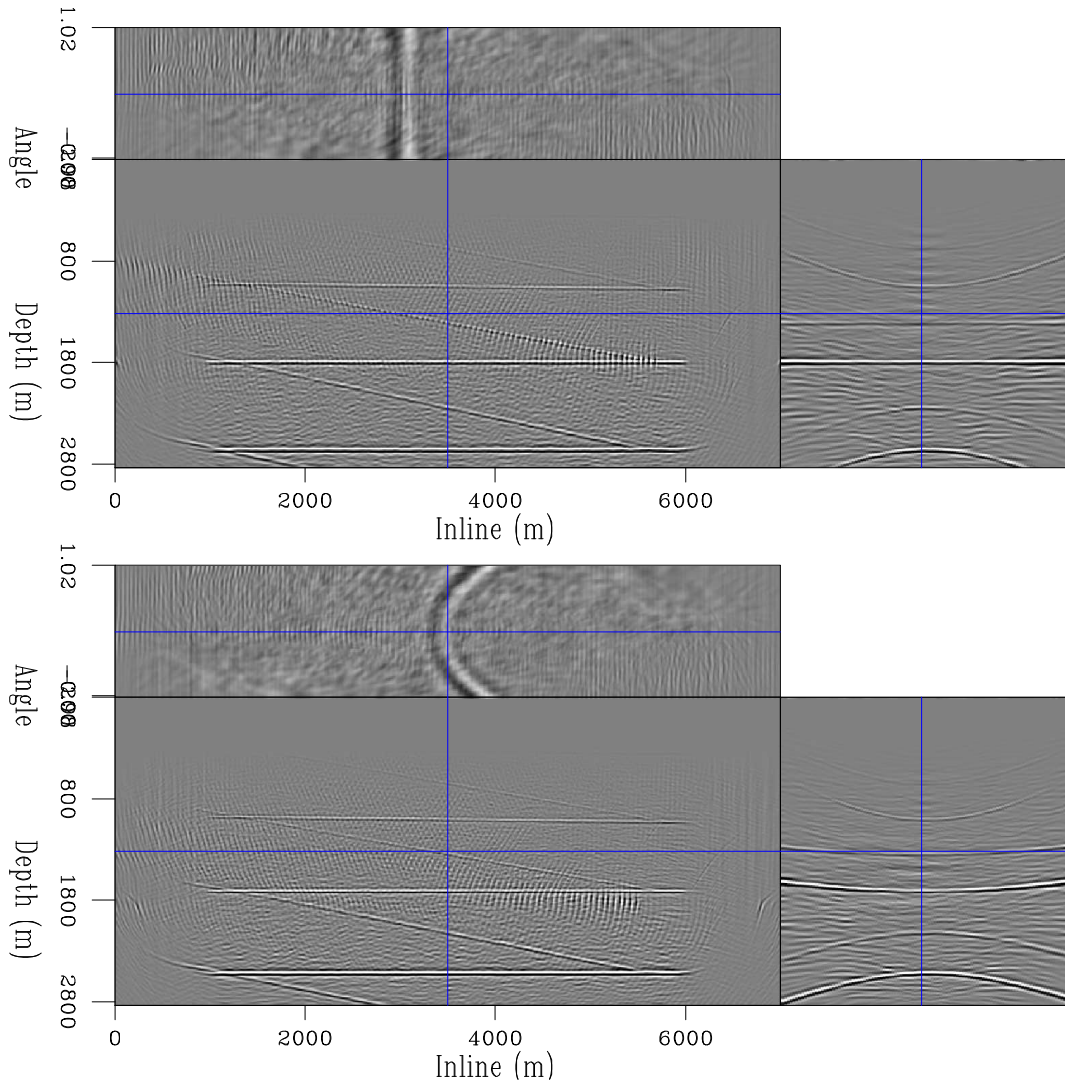

Figure 4.16: The same images as Figure 4.15 but with the third axis transformed into the subsurface angle domain, rather than subsurface offset. [**CR**] chap4/. angimgs

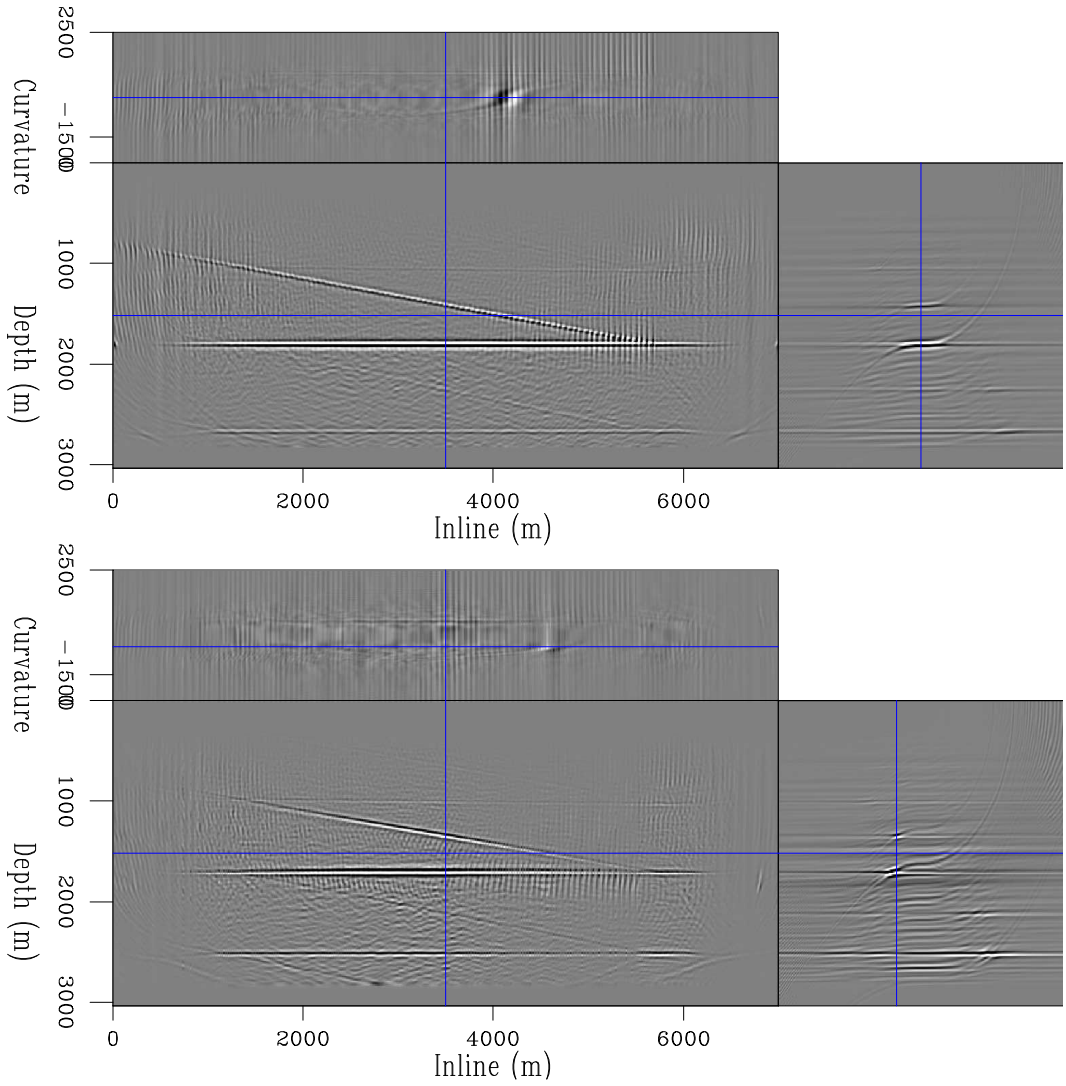

Figure 4.17: The angle domain image with the correct velocity model after a single parabolic transform (top) and the slow velocity image after a single parabolic radon transform. [**CR**] chap4/. anghypcomp

events have not been well focused as function of their curvature, and there is a lot of contamination across this space. It is possible to extend this domain transformation to an inverse process, and the result is shown in Figure 4.18. After ten iterations of this process, coherent events are now well focused at a given measure of curvature. It would be possible to window these events and reverse these processes, resulting in a separated dataset.

This process has two key problems implicit in these assumptions. One, is that subsurface reflection events will exhibit a measurable, single-parameter curvature in the angle domain. The other, is that primary events will focus at a similar level of curvature.

The former of these can be quickly disproved by imaging a more complex dataset, with steep structures and discontinuous reflectors. Under these conditions, events in subsurface offset exhibit a far less parabolic nature. Attempts to transform these to the angle domain and isolate primary reflections according to their move-out yielded no success. This is not as severe a limitation for WEMVA, since not every reflector's curvature is needed to be measured in order to back-project a meaningful model update. However, for deblending the loss of reflector information is unnacceptable. Additionally, to build these angle domain panels a regular and dense acquisition geometry is desirable, and this is not a good requirement for a separation engine.

Secondly, the fact that in this example the two events focused at the same measure of curvature was because the velocity model was a scaled version of the true model. If this was not the case, these events would focus to different amounts of curvature. This is not as strict a limitation, since a predetermined range of curvatures could be retained, and significant data-separation still achieved. However, these limitatations coupled imply that curvature-based image-space separation will fail under a variety of conditions.

Instead, posing the separation problem as an inversion will be investigated, where the minimization is posed as a function of these separated data. Initially, how these data behave after demigration will be tested.

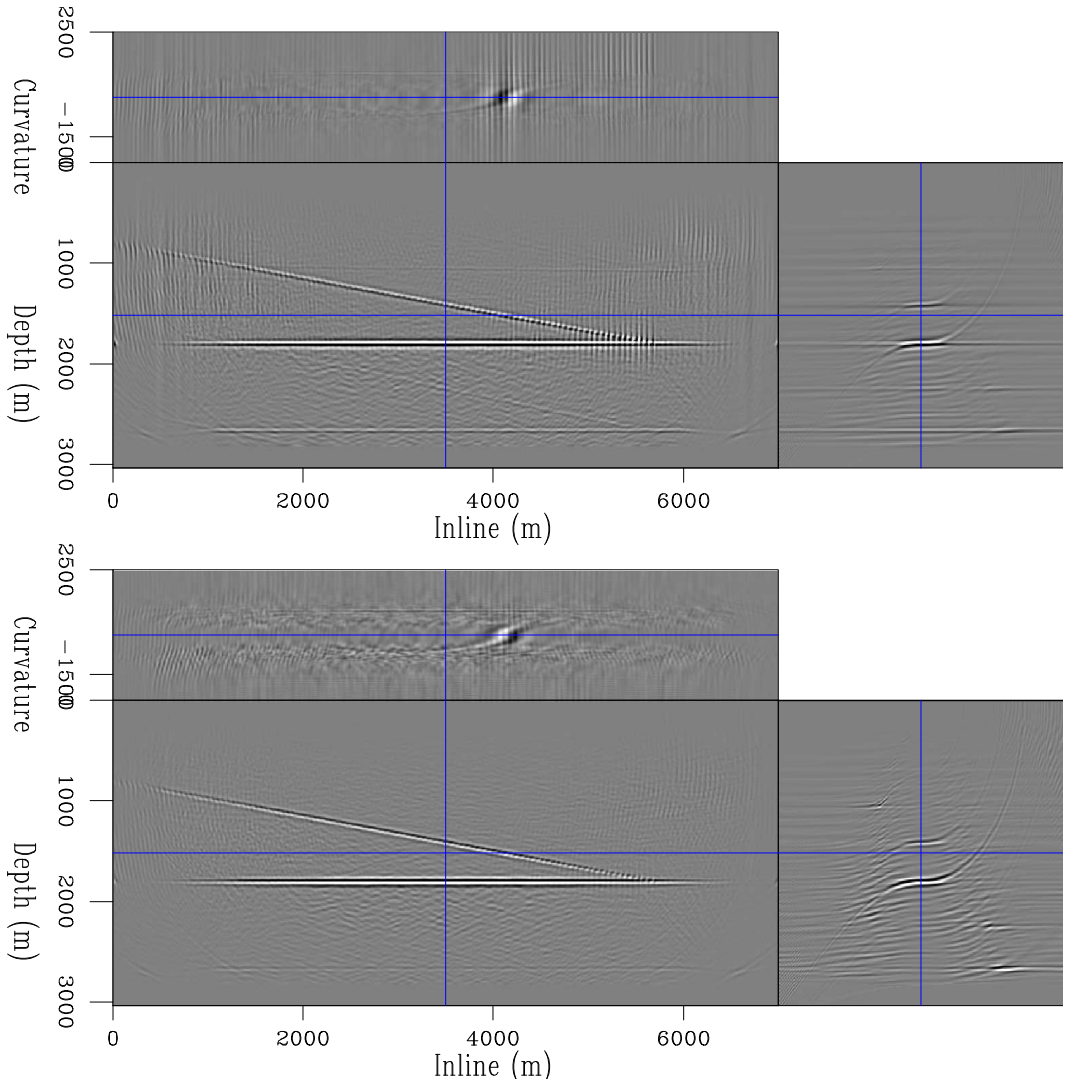

Figure 4.18: The angle domain image with the correct velocity model after a single parabolic transform (top) and after ten linear iterations of the transform (bottom.) Note the focusing at zero curvature (corresponding to a flat angle gather.) [**CR**] chap4/. anghyp

Adjoint demigration / Born modeling

As in Chapter 2, for linearized inversion, the adjoint procedure of equation 4.7 can be used to move back to the data domain. In this case, the forward procedure is now extended Born modeling. For Born modeling, during each imaging time step the receiver wavefield is convolved with the scattering estimate and then propagated. For extended Born modeling, during each imaging time step the receiver wavefield must now be convolved with the extended scattering estimate (this is the extended image.) This is, of course, both more computationally expensive, and requires additional memory. There are additional computation complications involved with extended Born modeling, when compared to extended RTM, and these are discussed at the end of Chapter 6.

To initially illustrate these processes, the same simple two-layer model will be used. This is instructive since all events are readily identifiable. The input, unblended data for these tests is the same as shown in Figure 4.3, and the correct and incorrect velocity migrations used for forward modeling are the results plotted in Figure 4.5 and Figure 4.6.

Applying the adjoint of this extended migration process gives the result in Figure 4.19, which resembles these input data fairly well. This result is informative of several attributes of this data recovery process. Amplitudes at early times and short offsets are under-represented; this is indicative of the need to extend the procedure to inversion.

Inverse Born modeling

While extended Born modeling results in the correct kinematic recovery of the input data, the amplitudes are inconsistent. Especially at early times and short offsets; this can be improved upon by extending forward modeling to an inverse procedure. Chapter 2 discussed linearised inversion at length, with the goal of recovering an accurate scattering model. What can be considered as the adjoint and forward procedures can

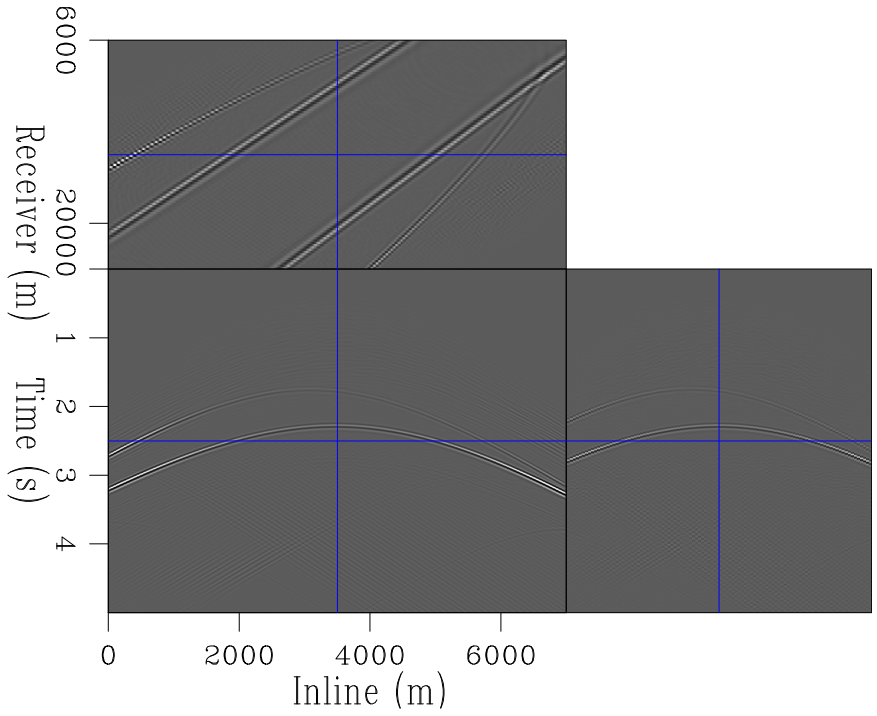

Figure 4.19: The ouput dataset from one pass of forward modeling, using the correct velocity model. [**CR**] chap4/. simpdemcv

be simply exchanged, and the laws of linear algebra and inversion are all still valid (Chauris and Benjemaa, 2010). Now, what was considered the model (the image) can be considered as the reference data, and what was previously considered the data (the recorded seismic reflections) are contained in what will be considered the model. The inversion scheme will act to recover the dataset that will migrate to provide the (cleaned) input image.

$$J(\mathbf{m}) = \|\mathbf{d} - \mathbf{L}\mathbf{m}\|_2^2 \tag{4.8}$$

$$J(\mathbf{d}) = \|\mathbf{m} - \mathbf{L}'\mathbf{d}\|_2^2 \tag{4.9}$$

Equation 4.8 shows the same objective function which was minimised in Chapter 2, where $\mathbf{L}$ is the Born modeling operator, $\mathbf{d}$ is the input data and $\mathbf{m}$ the scattering model. For inverse Born modeling, the objective function is formulated in equation 4.9. These will now be expanded into minimization algorithms to explicitly analyse the differences.

Algorithm 4 Extended linearized inversion

Calculate initial data-space residual $r = Em_0 - d$
while iter $<$ n_iter; iter++ **do**
  Create gradient $g_e = E'r$
  Create conjugate gradient $cg = Eg_e$
  Calculate step length
  Update m and data residual
end while
Output model

The algorithm for LSRTM is shown again in algorithm 4. Now, however, L has been replaced with E, to denote the extended imaging operator, and g with g_e, to clarify the increase in dimensionality. The size of the residual and the conjugate gradient are not changed. By switching the sense of the operators, data, model, residual and conjugate gradient, these scheme can be adapted to perform inverse modeling. This is summarized in algorithm 5.

Algorithm 5 Extended inverse Born modeling

Calculate reference extended image $i_e = E'd$
Calculate initial residual $r = E'd_0 - i_e$
while iter $<$ n_iter; iter++ **do**
  Create gradient $g = Er$
  Create conjugate gradient $cg = E'g$
  Calculate step length
  Update d_{out} and image-space residual
end while
Output dataset

Weibull and Arntsen (2014) used a similar procedure to remove multiples in the unfocused extended image space, and then used an inverse demigration scheme to recover amplitude consistent, multiple free data. A very similar methodology will be used in this thesis, in the context of data separation.

It is possible to approximate this extended operator, as shown in Hou and Symes (2015). This scheme can give an approximate, forward modeled dataset at a substantially lower cost than the full extended operator. However, in a strict inversion formulation it is not an exact adjoint to the imaging procedure. Such an approach could provie a valuable preconditioner to inverse Born modeling, and should be investigated.

Conventional, extended RTM of Figure 4.3 gave the image seen in Figure 4.5. In the previous section, a single pass of linearized forward modeling was applied and the output dataset analysed. For inverse Born modeling, this image can be considered the 'data' of the inversion - meaning it is a fixed reference, and is used to calculated the intial residual. Exactly the same procedure as Chapter 2 is run, but with the forward and adjoint operators swapped. The first gradient of the scheme (the adjoint result) is shown in Figure 4.19. It is immediately clear that the kinematics of the input data have been resolved, but the amplitudes are not correctly balance. As before, the error in the event positioning is incredibly low; the amplitude inconsistencies occur mostly at short offsets and early times.

After running inverse Born modeling for ten iterations the output dataset in Figure 4.20 is achieved. The amplitude inaccuracies are all fixed, and the error between these data and the input is very low. After only a couple of iterations the normalised residual is below 10%, and after ten iterations this measure is less than 0.03%.

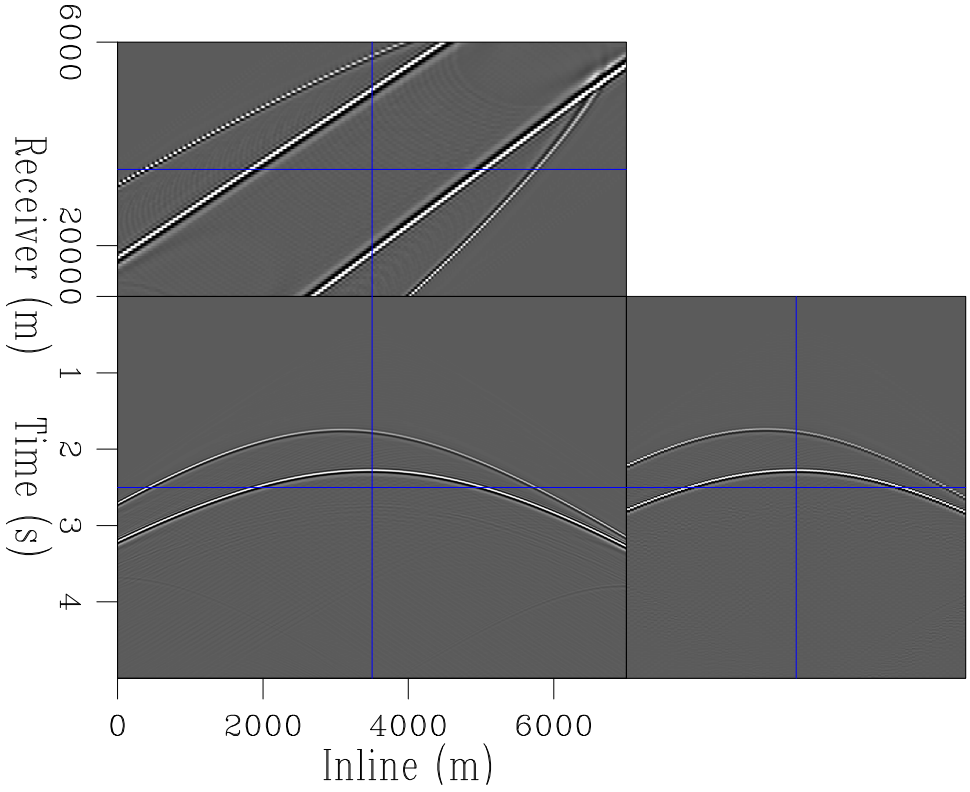

Figure 4.20: Inverse Born modeling using the correct velocity model, after ten iterations. [**CR**] chap4/. simpinvdemcv

To test the algorithm, a very inaccurate velocity model will now be used. The same constant velocity model will be used, which is inaccurate up to 20% in areas.

The first gradient (adjoint) result from incorrect velocity model extended demgration is in Figure 4.21. Similar to the exact velocity case, the kinematics are (largely) correct, however the amplitudes are not well balanced. This is exacerbated in the rough model case, and in addition to these amplitude inconsistencies, there are more artifacts induced. These arise, from the extended Born modeling. The recovered data after ten iterations are plotted in Figure 4.22. As before, the inversion has acted to

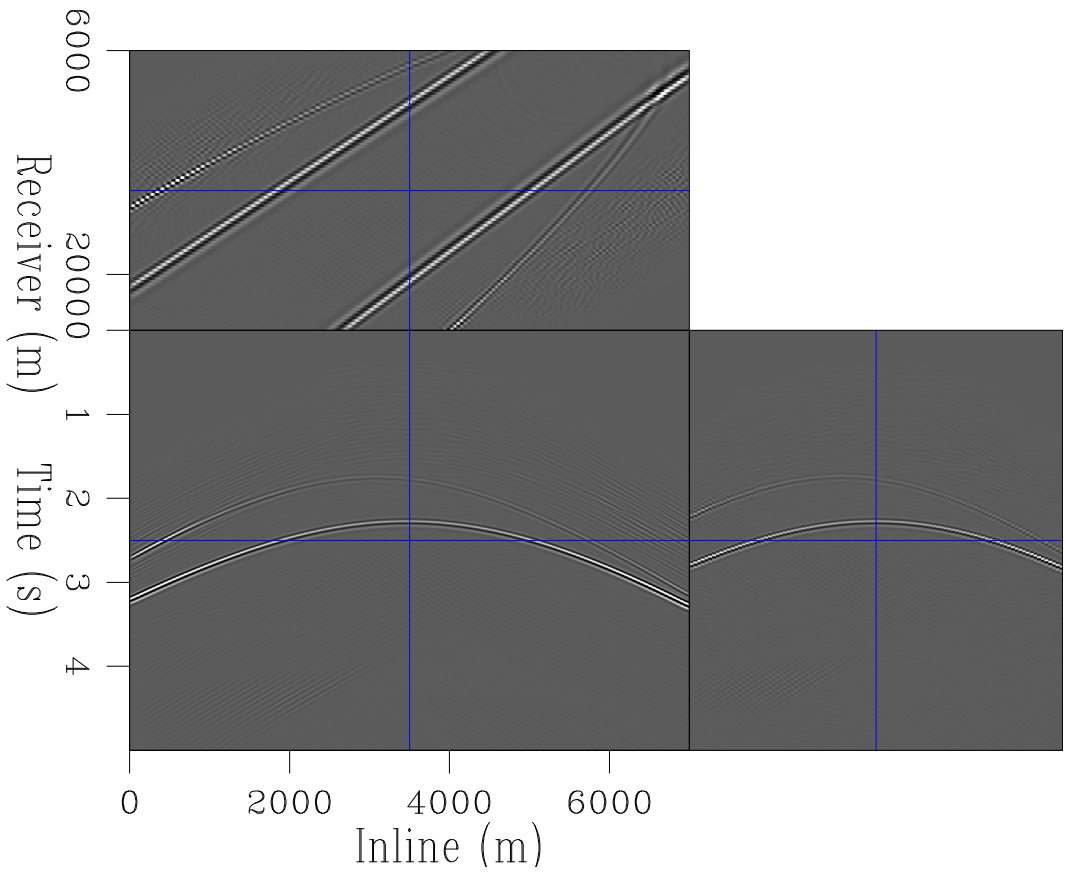

Figure 4.21: Adjoint Born modeling using an incorrect velocity model, after ten iterations. [**CR**] chap4/. simpdemincv

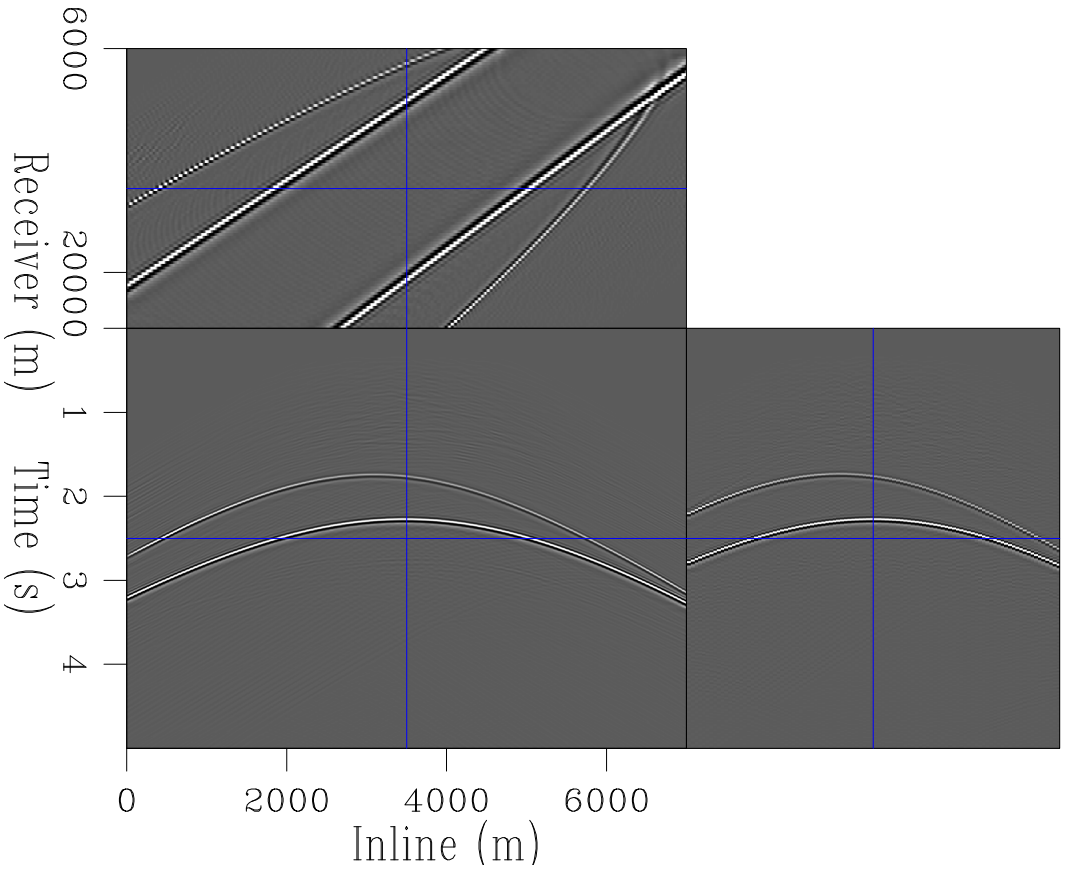

Figure 4.22: Adjoint Born modeling using an incorrect velocity model, after ten iterations. [**CR**] chap4/. simpinvdemincv

correctly balance these amplitudes and the output dataset closely resembles to input data, to within 0.3% error.

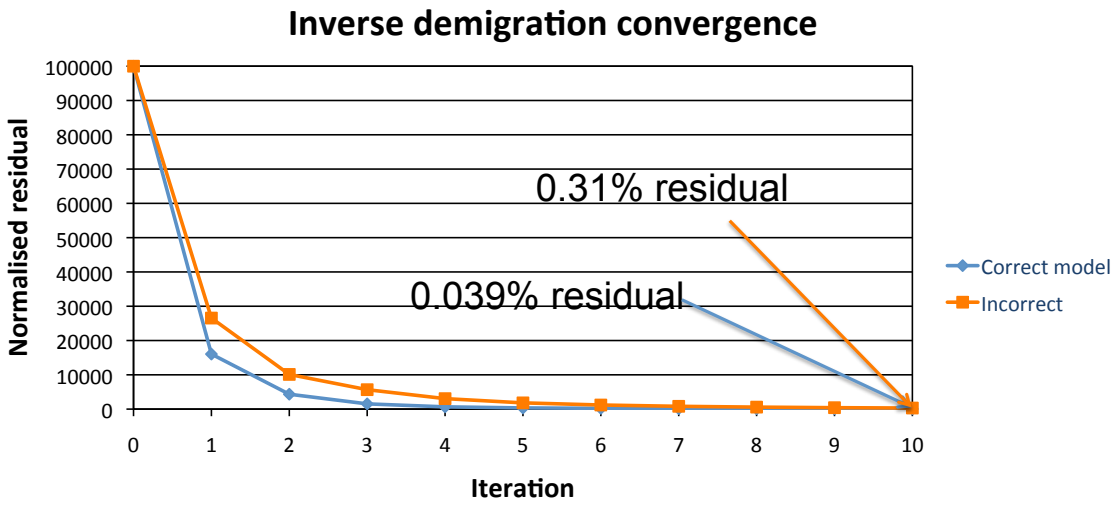

Figure 4.23: Convergence of inverse Born modeling as a function of iteration number. The normalised residual is measured in the image space. [**NR**] chap4/. simpinvconv

The convergence curve for both schemes are plotted in Figure 4.23. Unsurprisingly, the extended scheme is slower to converge than the exact model scheme. Again, within one iteration a residual of less than 10% is achieved, and after ten iterations these output data are correct to within 0.3%. This is a very acceptable level of error, particularly for a scheme which will be augmented with velocity updates.

These results demonstrate that extended inverse Born modeling can successfully recover input data, even after imaging with an incorrect velocity model. A similar approach can now be used, with the goal of data separation, rather than simple amplitude recovery.

IMAGE SPACE DEBLENDING

While primary blending is fixed post-acquisition, the style of proposed processing can influence how these data are initially acquired. Intuitively, the factors that will

influence these blended data are: the number of sources, the minimum recharge time for the airguns, the randomness of shot positionings, the randomness of shot timings, receiver geometry, and to a lesser extent, the number of receivers and the proposed recording length.

As mentioned earlier in this chapter, existing separation methods are critically dependent on the randomness of the source timings. Any induced repetition or predictability in these can induce debilitating artifacts (Abma et al., 2012). In the case of constant delays between shots, these methods entirely fail. For both flexibility, and confidence, in acquiring simultaneous data, it is desirable to relinquish these restrictions. The aforementioned three styles of shooting will be analysed in these upcoming sections: random delays, linear delays, and pseudo-linear delays.

Marmousi data separation

Figure 4.10, Figure 4.12 and Figure 4.14 show the extended images produced from these three datasets using the correct velocity. The differences in blending are manifested in the image space, although the coherency differences are not as pronounced as intuition may suggest. Even the linearly blended data becomes well dispersed in the image space. The artifacts are more coherent, but the focusing characteristics of the primary events, and the differences in contrast, suggest that separation should be possible.

The recovered datasets after adjoint separation can be seen in Figure 4.24, Figure 4.25 and Figure 4.26. These can be contrasted with the reference, unblended dataset, in Figure 4.8. Each of the blending schemes have been well separated and the resultant datasets could be used for conventional velocity estimation and imaging. Again, amplitudes at early times and shot offsets are weaker than in Figure 4.8, which should be the result if the separation was exact. This can be improved upon by using the inverse scheme.

It should be noted that there are some fractionally more coherent artifacts in the recovered data from the linear encoding. Nonetheless this methodology separated

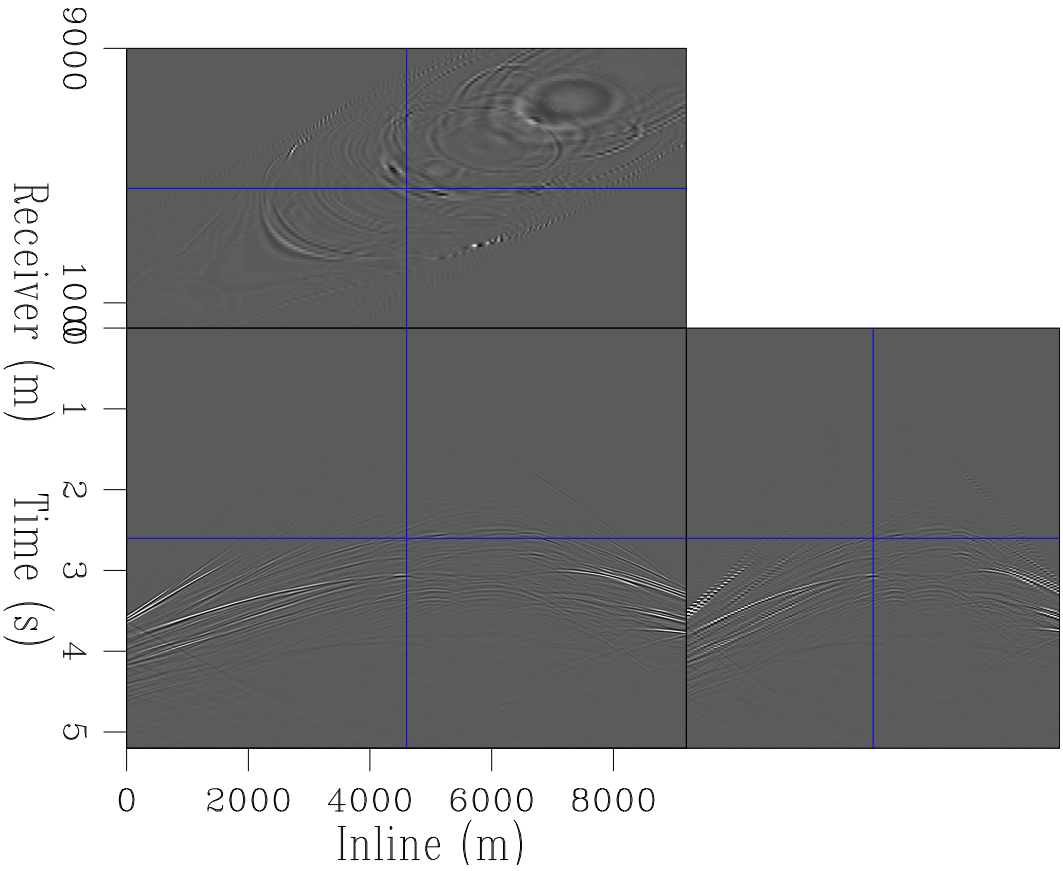

Figure 4.24: The output dataset after applying one pass of Born modeling to Figure 4.10, which was the image created from a randomly delayed dataset. [**CR**] chap4/. marmrndblndidem

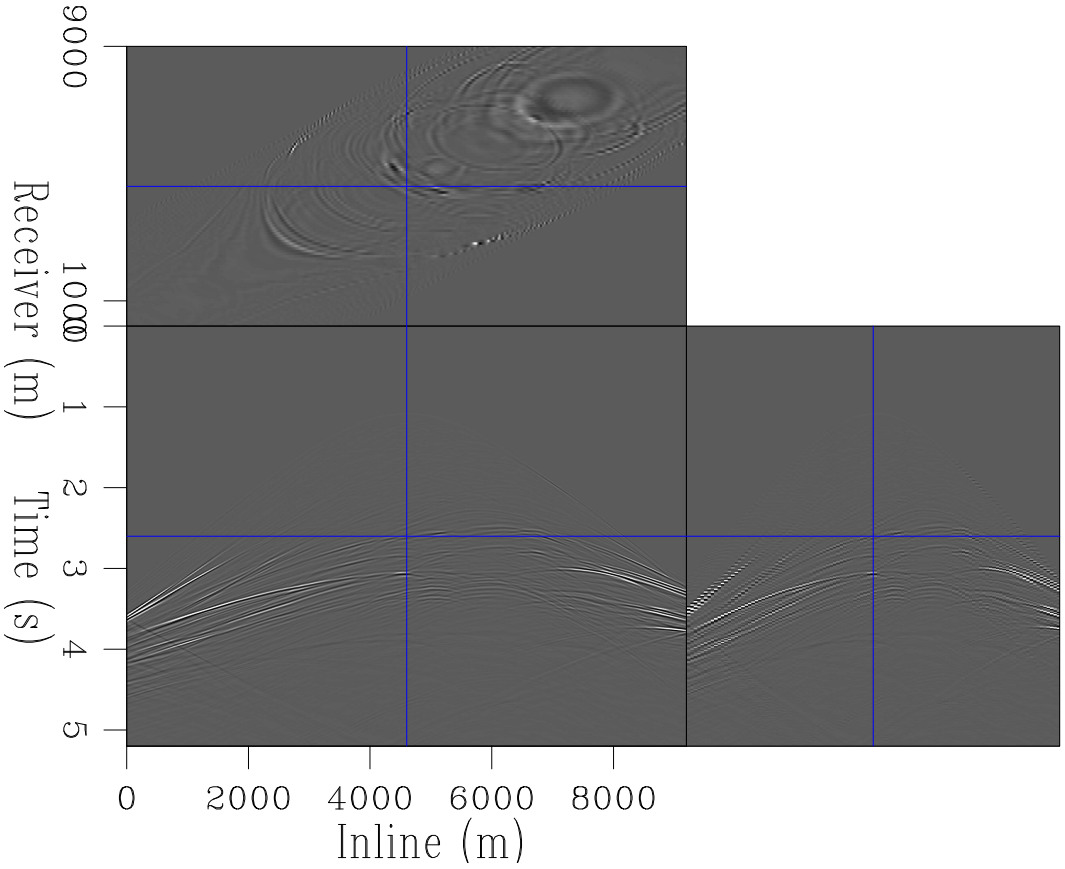

Figure 4.25: The output dataset after applying one pass of Born modeling to Figure 4.12, which was the image created from a linearly delayed dataset. [**CR**] chap4/. marmlinblndidem

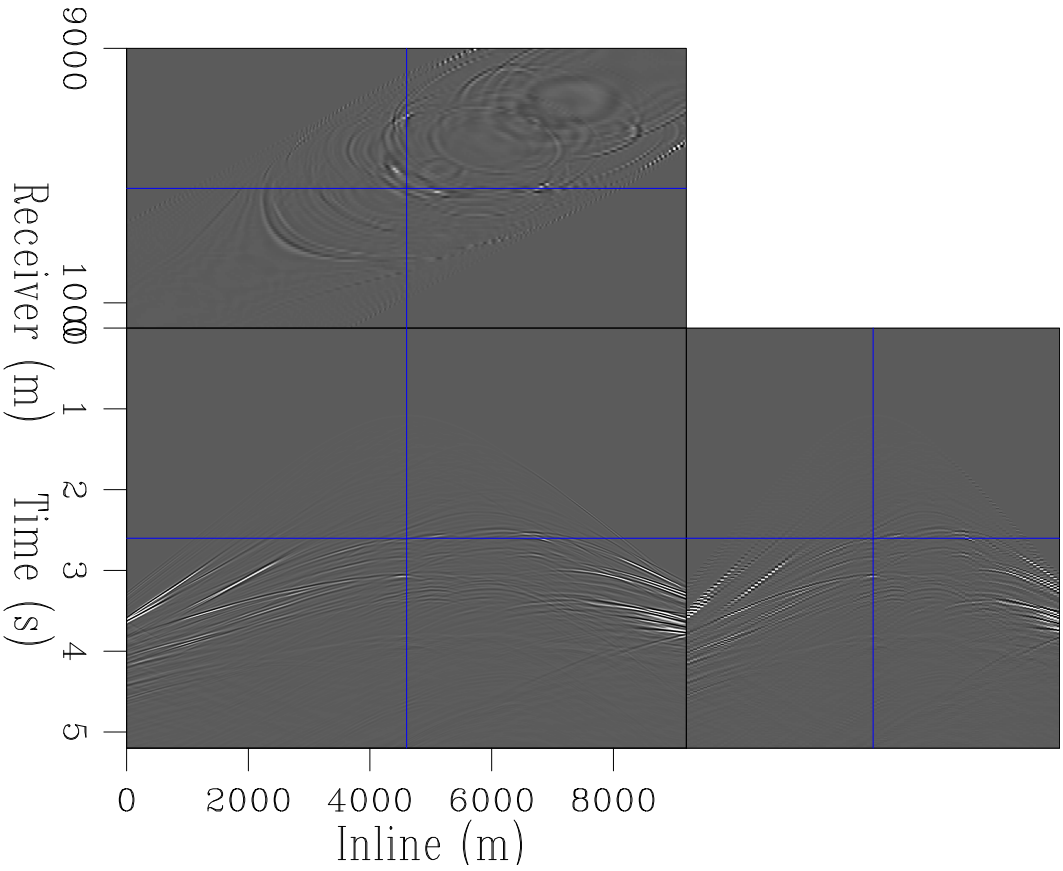

Figure 4.26: The output dataset after applying one pass of Born modeling to Figure 4.14, which was the image created from a pseudo-linearly delayed dataset. [**CR**] chap4/. marmplinblndidem

these linearly delayed data very accurately.

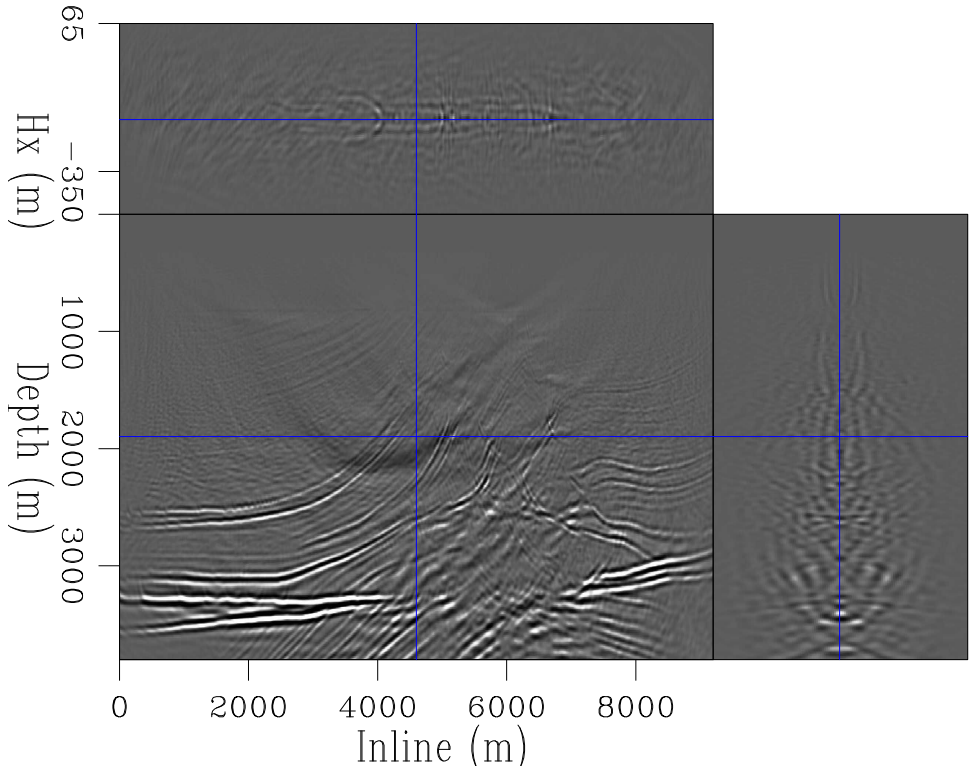

Figure 4.27: The extended image from migrating Figure 4.13 (data with constant delays) using a rough velocity model. [**CR**] chap4/. marmplinblndiminc

Figure 4.27 shows the result from migrating these data, but using an inaccurate velocity model. The primary events are now not well focused at zero-subsurface offset, and the focusing contrasts between primary and secondary events is far less. Energy of interest now spans many of the acquired subsurface offsets and distinguishing events from primary and secondary energy is less obvious. Separation as an inverse problem was not run on these data, a more complicated model was used instead.

Using data acquired over the Marmousi model gives informative results about the suggested procedure, but to further confidence over this methodology a more difficult example must be used. A section of the SEAM model was windowed, featuring rugose reflectors, continuous reflectors and a steep salt body with carbonate top and sedimentary inclusions. These attributes comprise many of the difficulties of

contemporary imaging targets. If this separation method can perform well over these data then much stronger conclusions can be made.

Blended inversion

Adapting linearised inversion to inverse forward modeling involved reversing the operators, and the sense of what was the model and what was the data. Creating a blended inversion scheme to recover separated data involves an extra step, since the blending must be accounted for.

$$\mathbf{d}_b = \mathbf{\Gamma}\mathbf{d}_s \tag{4.10}$$

If $\mathbf{d}_s$ denoted the separated data, $\mathbf{d}_b$ the blended data, and $\mathbf{\Gamma}$ the operator that creates these blended data (applies the time shifts), then the blending can be described as equation 4.10. This operator, $\mathbf{\Gamma}$, contains all the necessary time shifts to move between the domains, and takes an continuous, input dataset. It windows this input dataset according the source times and a desired recorded length, and outputs a series of shot gathers. Of course, these shot gathers will contain blended contamination, since individual shots have simply been windowed and no separation has been attempted. This process of windowing and aligning is sometimes referred to as pseudo-deblending.

$$\mathbf{m} = \mathbf{L}'\mathbf{\Gamma}'\mathbf{d}_b \tag{4.11}$$

$$\mathbf{m}_e = \mathbf{E}'\mathbf{\Gamma}'\mathbf{d}_b \tag{4.12}$$

As earlier, $\mathbf{L}'$ can represent the zero-offset imaging operator, and $\mathbf{E}'$ the extended imaging operator. The input, blended data can be cut and aligned, using $\mathbf{\Gamma}$, and then migrated, using either operator. This results in either a zero-offset image or an extended image, shown in equation 4.11 and equation 4.12 respectively. These images

can then be used for forward modeling, or for inverse Born modeling. Cost functions, similar to those in Chapter 2, can then be minimized. Now, the function will aim to reduce the misfit related to $\mathbf{d}_s$, the separated dataset.

$$J(\mathbf{d}_s) = \|\mathbf{L}'\mathbf{d}_s - \mathbf{m}\|_2^2 \tag{4.13}$$

$$J(\mathbf{d}_s) = \|\mathbf{E}'\mathbf{d}_s - \mathbf{m}_e\|_2^2 \tag{4.14}$$

A similar algorithm to algorithm 5 is then used; the new zero-offset and extended objective functions to be solved are formulated in equation 4.13 and equation 4.14, and this adapting algorithm expanded as algorithm 6. These new objective functions are extensions of equation 4.9, with $\mathbf{\Gamma}$ included. By making this extended image the input for an inverse Born modeling scheme, these data can be effectively separated into individual shot records.

Algorithm 6 Inverse deblending

Calculate reference extended image $i_e = E'\Gamma'd$
Calculate initial residual $r = E'd_0 - i_e$
while iter $<$ n_iter; iter++ **do**
  Create gradient $g = Er$
  Create conjugate gradient $cg = E'g$
  Calculate step length
  Update d_{out} and image-space residual
end while
Output dataset

To demonstrate the effectiveness of minimising equation 4.14 for shot separation, a model and dataset more complex than Marmousi will be used. A same section of the SEAM model that was used in Chapters 2 and 3 was used, since this features steep deeps, high velocities, and a range of scattering contrasts. While the Marmousi tests were informative, it does not feature these final two attributes.

As a reminder, the velocity model used for simulating the comparison, unblended data is shown in Figure 4.28, and the reference dataset can be seen in Figure 4.29. A

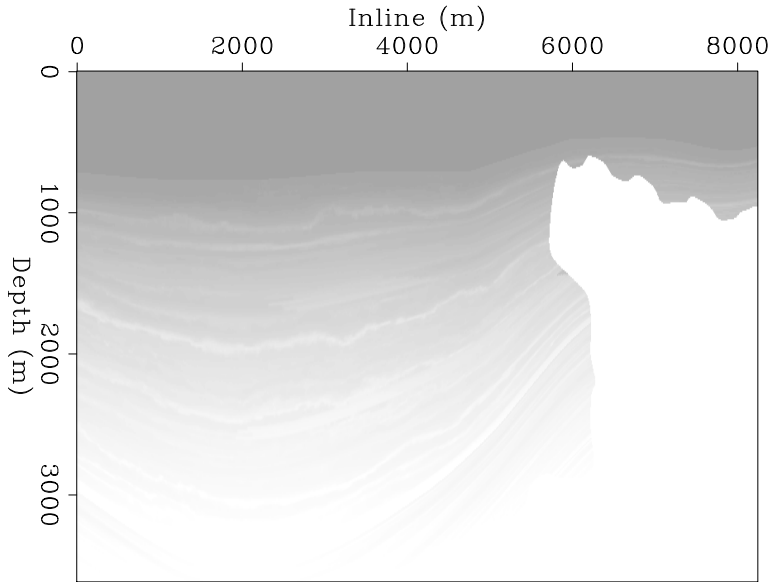

Figure 4.28: The velocity model windowed from the SEAM model, used for a more realistic separation test. [**ER**] chap4/. seamvel

pseudo-linear blending scheme was implemented, since this is the most realistic and can pose separation problems. These data after blending are plotted in Figure 4.31.

An approximate, unfocused image was created in the extended image space by massively smoothing the input velocity model, creating some regions where the velocity is approximately, and some regions where the velocity incorrect up to 15%. This is more realistic than simply scaling the full model, and will create a very loosely focused salt boundary, creating a valid separation test. The inaccurate model used is plotted in Figure 4.30, and the loosely focused, noisy image after migration, in Figure 4.32. After ten iterations of this inverse deblending process (algorithm 6), the output, separated dataset is shown in Figure 4.34. Through comparison to the input, blended dataset, this scheme has been very successful in separating these shots into uncontaminated gathers. No remaining energy from the interfering data remain in these separated shots, however a few, minor artifacts from extended modeling remain.

Figure 4.35 shows the results of applying RTM to Figure 4.29, Figure 4.33 and Figure 4.34 respectively. It is clear that all images are directly comparable, although a

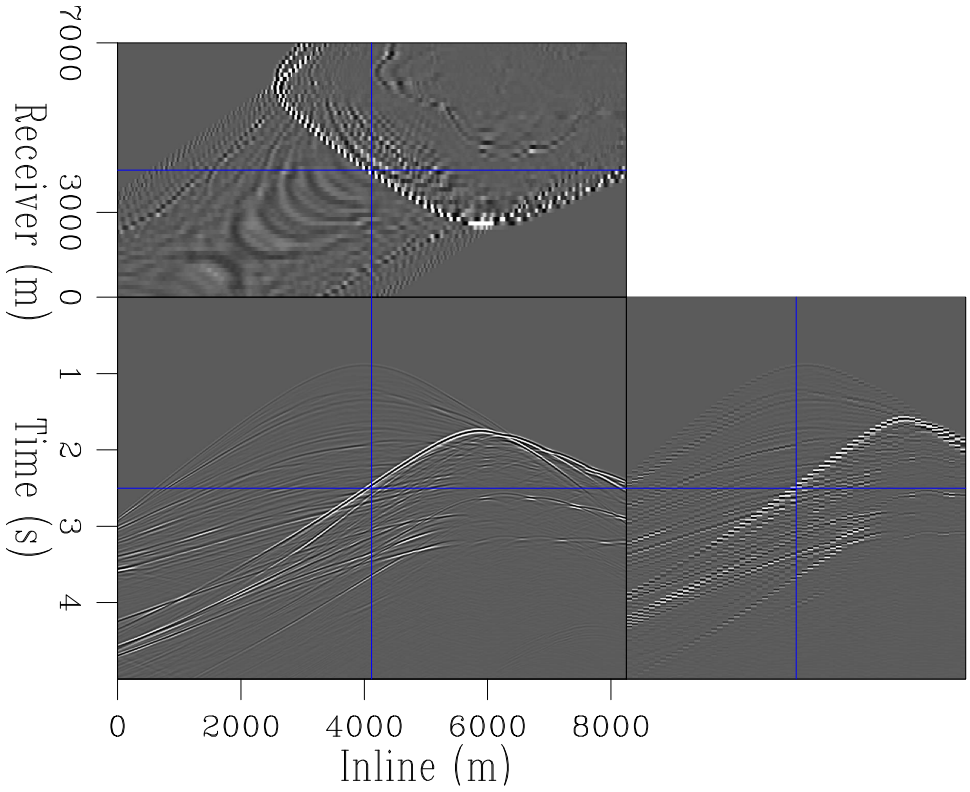

Figure 4.29: A conventional dataset acquired using the section of the SEAM model. [**CR**] chap4/. seamdata

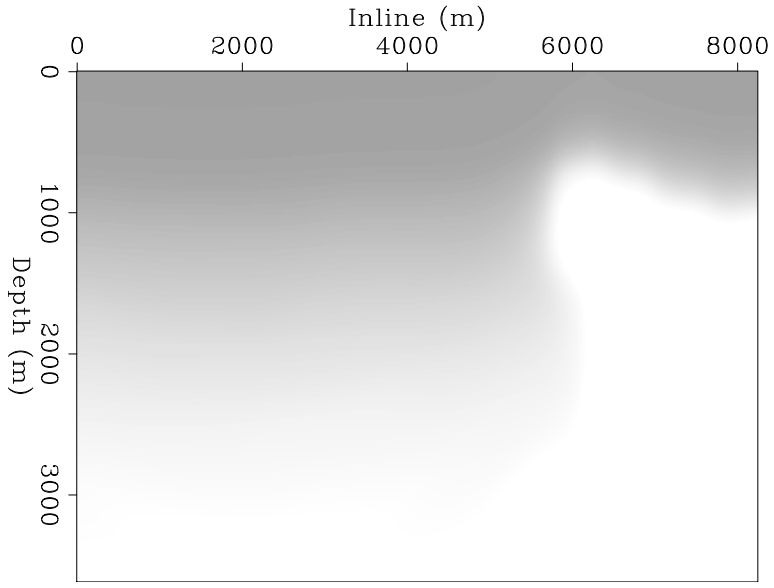

Figure 4.30: A heavily smoothed version of Figure 4.28, used for inaccurate model separation. A rectangle filter of 30 model points x 30 model points was used. [**ER**] chap4/. seamvs

small amount of noise is noticeable within the salt body in the two lower images. The salt boundary is slightly less smooth in the image using the data separated with the inaccurate model, however this is the only obvious imperfection. Using the heavily smoothed model does not appear to have resulted in the loss of any information.

Extended images produced by migrating 30 shots before and after deblending are shown in Figure 4.36 and Figure 4.37, using the rough migrating velocity. Due to the inaccurate model, both images are unfocused and noisy, but the image after deblending is substantially cleaner, and would be easier to perform preliminary interpretations on, as well as moveout based estimations. Often for quick velocity updates / QC checks a subset of the data will be used, and this demonstrates the effectiveness of performing separation first.

The convergence to the input dataset is shown in Figure 4.38. This was calculated by reblending the output data, and comparing to the input blended data. Since the only input is the blended data, $\mathbf{d}_b$, then to measure separation these output

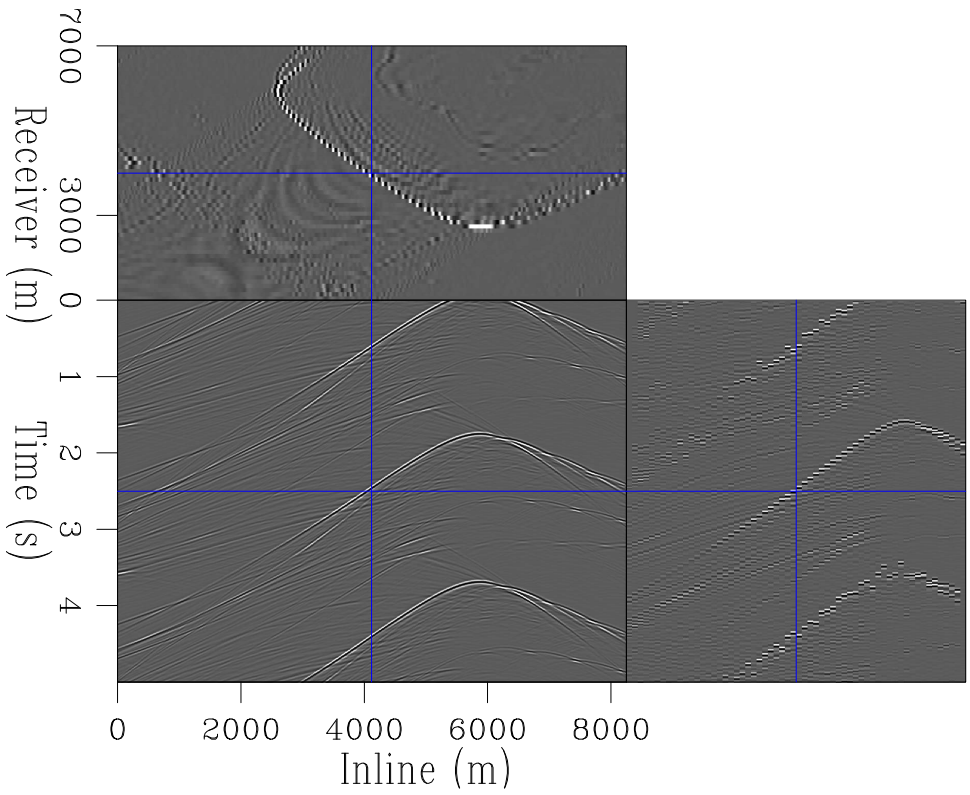

Figure 4.31: A pseudo-linearly blended dataset acquired using the section of the SEAM model. [**CR**] chap4/. seamdatain

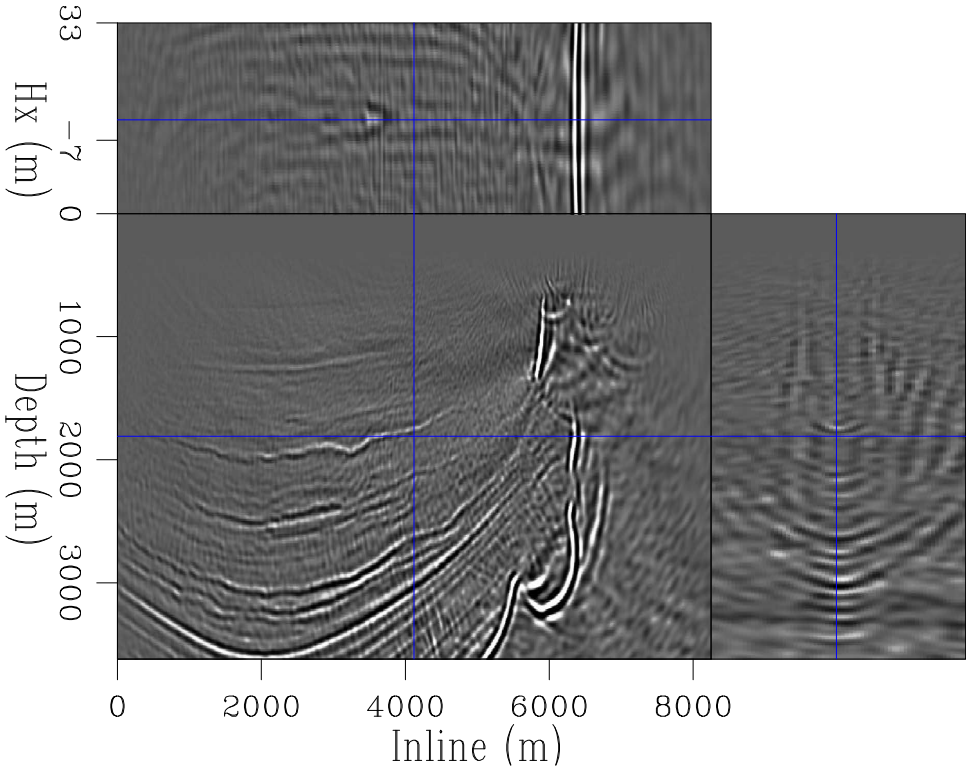

Figure 4.32: The extended image created by migrating the blended dataset with the velocity model shown in Figure 4.30. [**CR**] chap4/. seam2dextimg

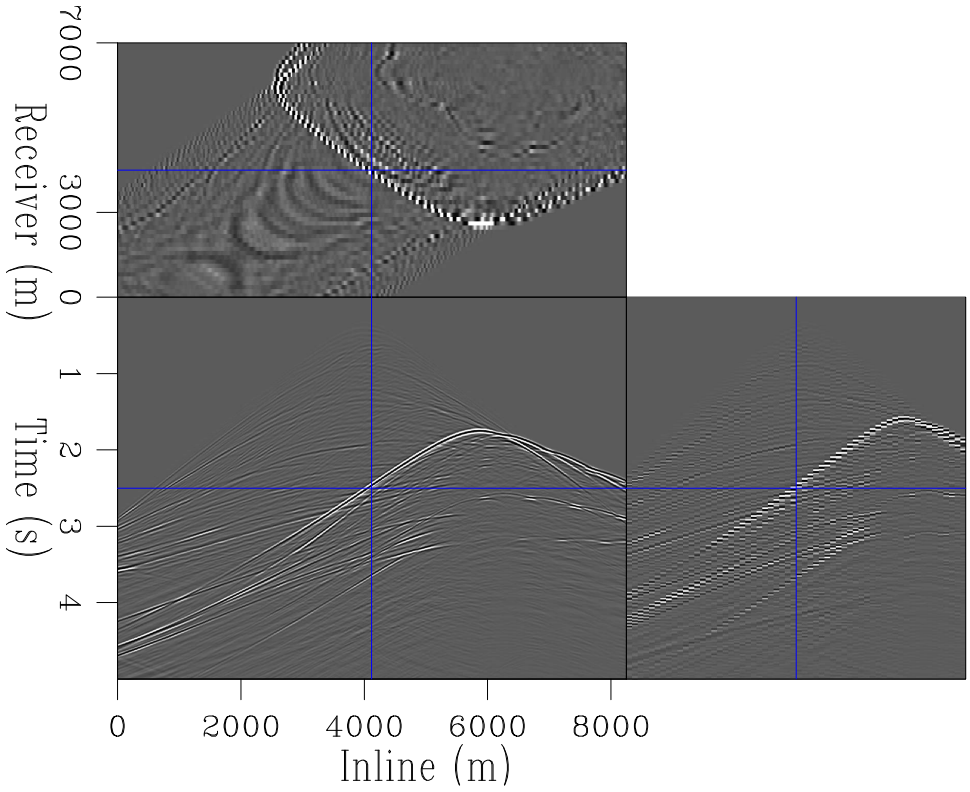

Figure 4.33: The output separated data after 10 iterations of inverse deimgration using the correct velocity model. [**CR**] chap4/. seamcdataout

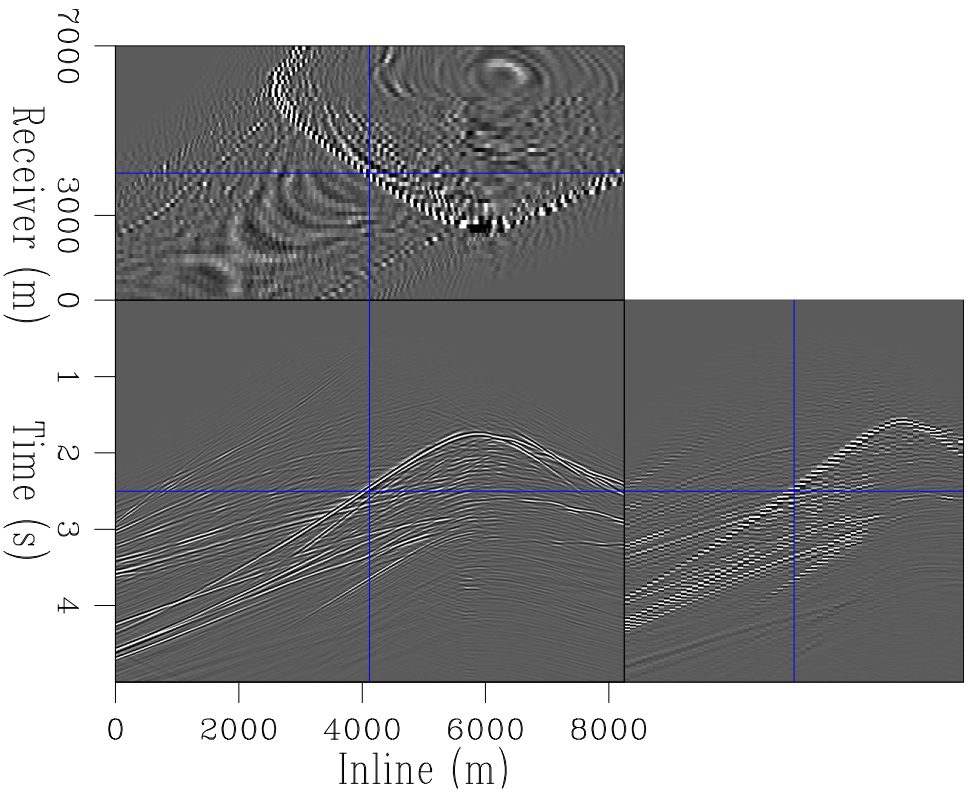

Figure 4.34: The output separated data after 10 iterations of inverse deimgration using an inaccurate velocity model. [**CR**] chap4/. seamdataout

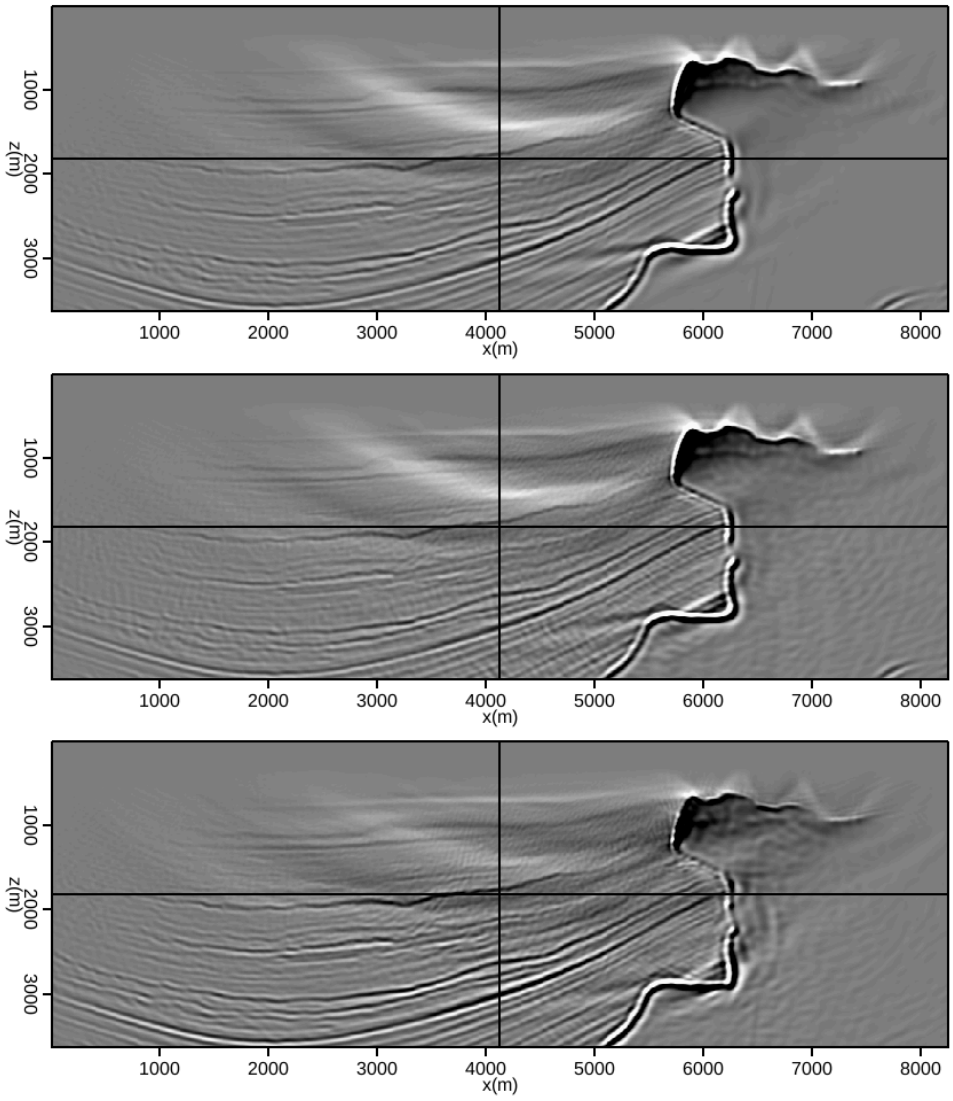

Figure 4.35: Images created from conventionally acquired data, blended data separated using the correct velocity, and blended data separated using an inaccurate velocity model respectively. chap4/. imagesafterseparation

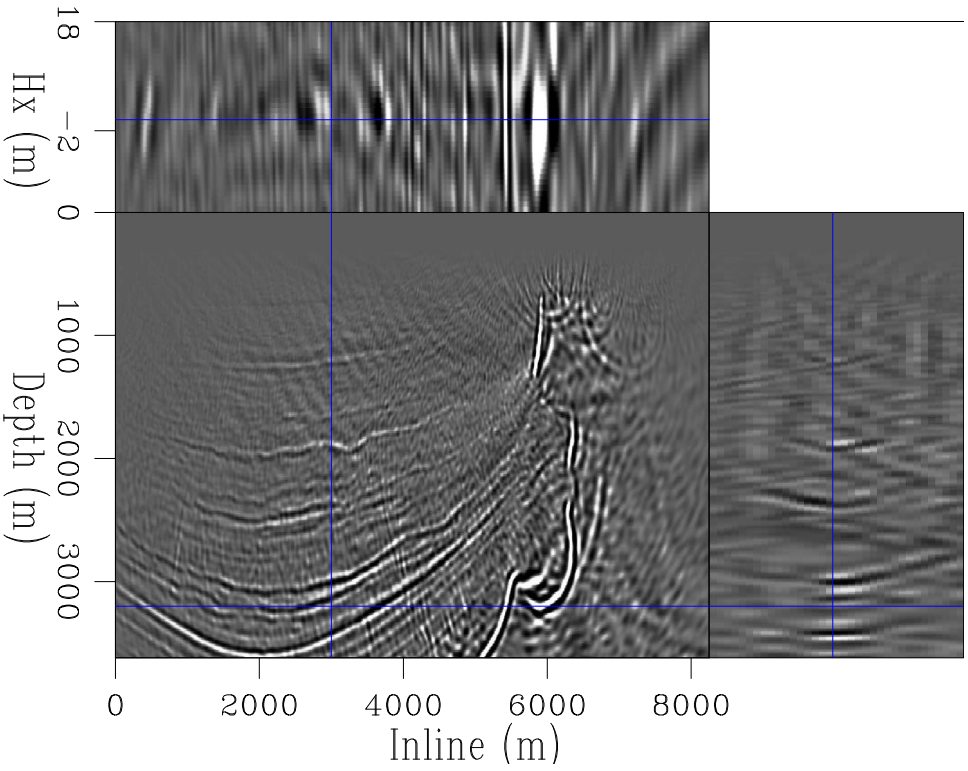

Figure 4.36: Thirty shots migrated into the extended domain before deblending, using the inaccuarte separation velocity. [**CR**] chap4/. seamextpresep

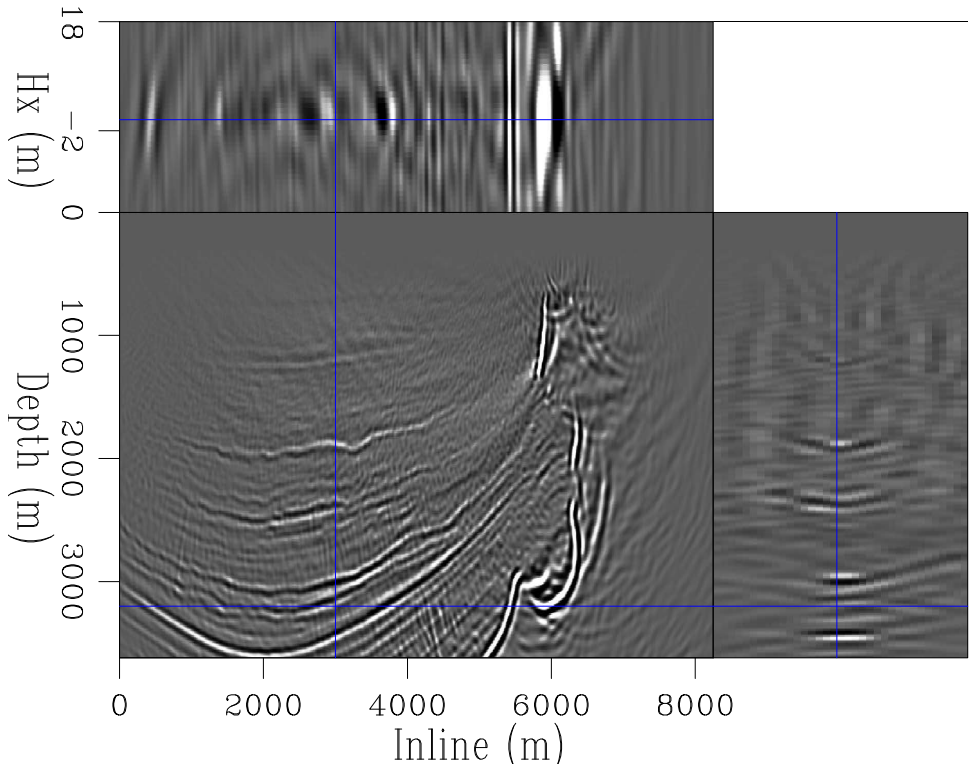

Figure 4.37: Thirty shots migrated into the extended domain after deblending, using the inaccurate separation velocity. [**CR**] chap4/. seamextpostsep

data, $\mathbf{d}_s$, should be reblended, and the output compared to $\mathbf{d}_b$. Other quantitative measures could be misleading, so $\|\mathbf{d}_b - \mathbf{\Gamma}\mathbf{d}_s\|_2^2$ is plotted. The system deblends the data almost immediately, and requires iterations to recover the amplitude content of the input data, and to reduce modeling artifacts. This process does not converge to as low a residual as unblended inverse linearized forward modeling (Figure 4.23), but considering how much more complex the model and data were for this SEAM test, it does a very comparable job.

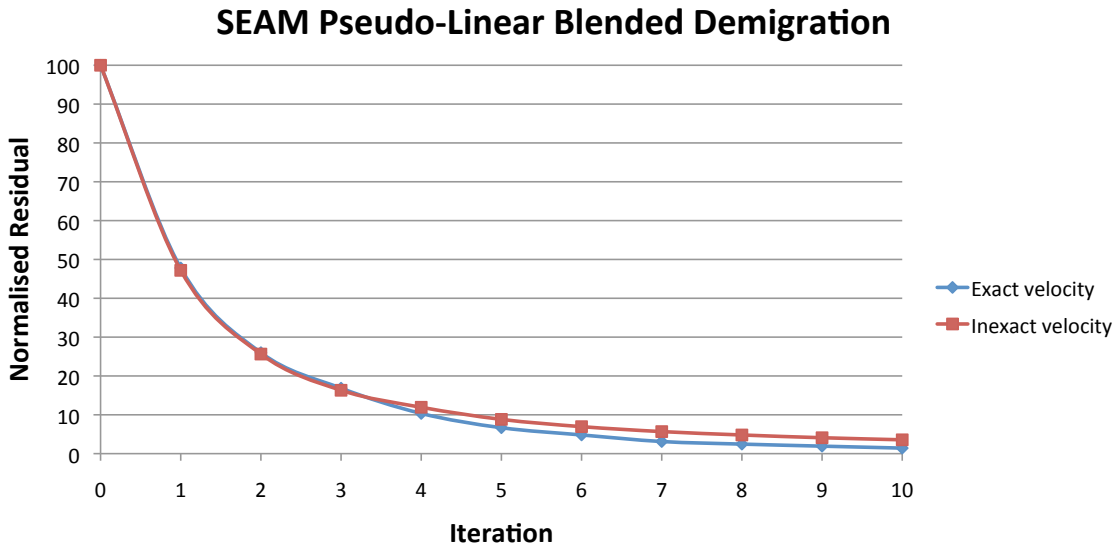

Figure 4.38: How the separation algorithm performs, for SEAM, as a function of iteration number. This normalised residual is the L2 norm of the difference between the input, blended data, and the output, separated data after reblending. [**NR**] chap4/. seamconv

CONCLUSIONS

Primary blending poses many more processing and imaging challenges than secondary blending, due to the fact the the blending is fixed. For field blended data to integrate well into existing processing and imaging work flows, the ability to separate these data into their conventionally acquired equivalent dataset is desirable.

The image space provides a powerful transform for this separation, since even with constantly delayed data many high amplitude blending-related artifacts are stacked out, and for randomly blended data the separation is provided almost immediately. By formulating the imaging in the extended image space a good velocity model requirement can be relaxed, since all kinematic and amplitude information for the primary data is saved, and the overlapping data is still massively reduced in both focusing and moveout characteristics.

For simple, layered models it is possible to isolate these overlapping data as a function of their curvature, however for more complex data this is not possible. Instead, by fixing this extended image and posing data recovery as an inverse, extended Born modeling problem, a separated dataset with comparable amplitudes and correct kinematics can be constructed.

Chapter 5

Field data example

The results thus far have been created using synthetically generated data. Whilst this is a useful exercise to validate and test concepts, it is necessary to demonstrate how this approach works on a 3D field data set. Field data features a range of attributes that were missing from these synthetic data, including converted waves, incoherent noise, coherent noise, processing artifacts, irregular sampling of receiver and source points, anisotropy, attenuation, and more. Each of these will pose new challenges for the process of inverse Born modeling, and the procedure must be robust enough to handle these real-world imperfections.

THE DATASET

The dataset used was acquired using Ocean Bottom Node (OBN) acquisition, will a total of 1195 receivers. Each receiver had a maximum offset (source position range) of 20 km in both inline and crossline directions, and the time sampling was 4 ms. The full imaging cube was 30 km x 30 km x 18 km, although for the purpose of demonstrating image-based shot separation, a smaller section was windowed.

An area of 20 km x 20 km x 8 km was targeted for imaging, using 103 receivers, which featured a concurrent shooting pattern. From the principal of reciprocity, these

can be considered as 103 shot gathers. Initially, RTM and linearized inversion were applied to these gathers. The benefits of this were twofold: the concept of LSRTM could be tested on field data, and a reference image to compare post-separation results and be created. The nature of both the Earth models provided and the acquisition necessitated several changes to the imaging approach from the last three chapters.

Firstly, the survey area featured a variety of dips, salt bodies, and subsurface stress states. This meant that earlier efforts at imaging the target described the subsurface as a Tilted Transverse Isotropic (TTI) medium (Tsvankin and Thomsen, 1995). The previous chapters stated that the subsurface could be described using a velocity field, $\mathbf{v}(x, y, z)$, which could be split into high-wavenumber and low-wavenumber components. This assumed an acoustic, isotropic subsurface (discussed in Chapter 2.) For a TTI medium, four additional parameter fields (as well as velocity) are needed to simulate wave propagation and perform imaging. There is some flexibility with the exact parameterisation; however the method herein used two Thomsen's anisotropic parameters (Thomsen, 1986), ϵ and δ, as well as two angular fields, ϕ and θ, describing dip and azimuth. These are each represented on the same gridded volume as the velocity field, resulting in significant additional necessary memory allocation. In addition to the memory required for allocating the Earth models, more wavefields must be allocated to accurately track these waves, as well as a more intense computational kernel.

When compared to acoustic propagation, TTI uses roughly four times the memory, and is overall around five times slower. This will elaborated upon at the end of Chapter 6. A propagation engine similar to those proposed in Alkhalifah (2000) and Zhang et al. (2003) was used. Such methods have been demonstrated to work on migrating field data using RTM by Fletcher et al. (2009).

Secondly, these data had been processed into down-going receiver gathers, which is typical for OBN imaging (Grion et al., 2007). This approach uses the first sea-surface related multiple for imaging, since this provides a significant increase in aperture for imaging, with no reduction in signal (since the sea surface is almost a perfect reflector) (Dash et al., 2009). A schematic detailing the difference between upgoing

and downgoing signals for OBN acquisition can be seen in Figure 5.1. For practical imaging, this is simulated by doubling the water column above the model, and using the sea surface as a mirror - placing the source in this new layer (Wong, 2014). This approach correctly simulates the down-going wavefield, without any up-going events, and is shown in Figure 5.2.

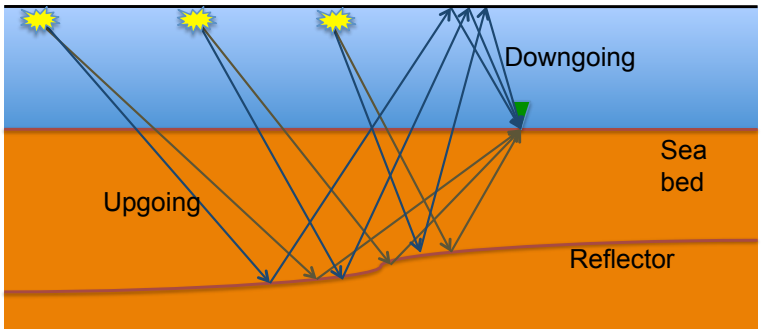

Figure 5.1: A diagram of upgoing and downgoing rays for OBN acquisition. [**NR**] chap5/. allgoing

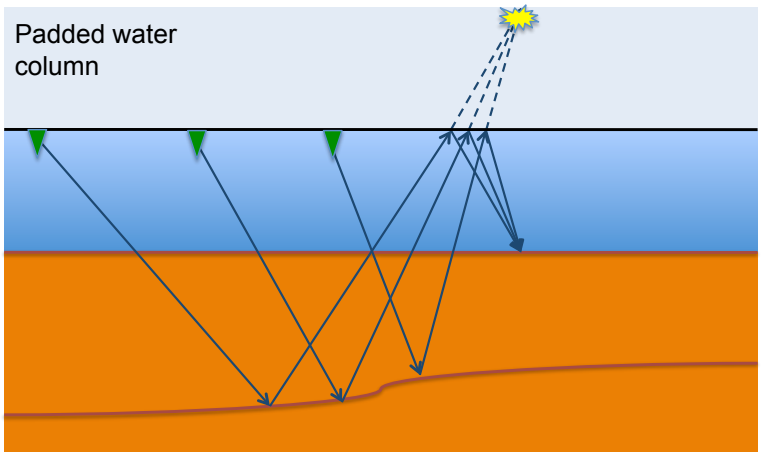

Figure 5.2: A diagram of how only downgoing rays can be simulated, and the principle of reciprocity. [**NR**] chap5/. mirroracq

TTI IMAGES

Figure 5.3 shows a rotated section of the image produced using TTI RTM on these 103 down-going receiver gathers. The inline aperture (in terms of source positioning) was greater than the crossline, and this is noticeable. Several salt bodies are identifiable, with well resolved edges, and a variety of reflectors have also been imaged. These events align well in both depth and lateral positioning with a provided reference image. Figure 5.5 then shows this section after Automatic Gain Control (AGC) has been applied. This is a useful tool on occasion, since it uses a rolling window to artifically boost energy in poorly illuminated areas, and this demonstrates that there are coherent reflectors throughout the entire model. Although there is a significant quantity of imaging induced noise.

Next, Figure 5.4 shows this same section after five iterations of linearized inversion. The inversion result after AGC is also shown, in Figure 5.6. There are numerous improvements, but the most pronounced benefits are seen around the salt bodies. The edges and bottoms of these salt bodies are very sharply resolved, with subsalt reflectors becoming more coherent and higher in amplitude. Many contemporary imaging targets are sub-salt, and so a procedure which can improve sub-salt resolution is oft sought after.

Other improvements are the general balancing of reflector amplitudes with depth, improved overall coherency of events, higher wavenumber boundaries, and a reduction in imaging related artifacts. It should also be noted that the inversion does struggle in some areas - particularly shallow sediments away from the source lines along the inline direction. This is due in part to the limited aperture in these directions, and due to salt-related refraction events, incorrectly included during imaging.

Nonetheless, given a very large, undersampled model and a relatively sparse set of receivers, the improvements of inversion over adjoint imaging are very encouraging.

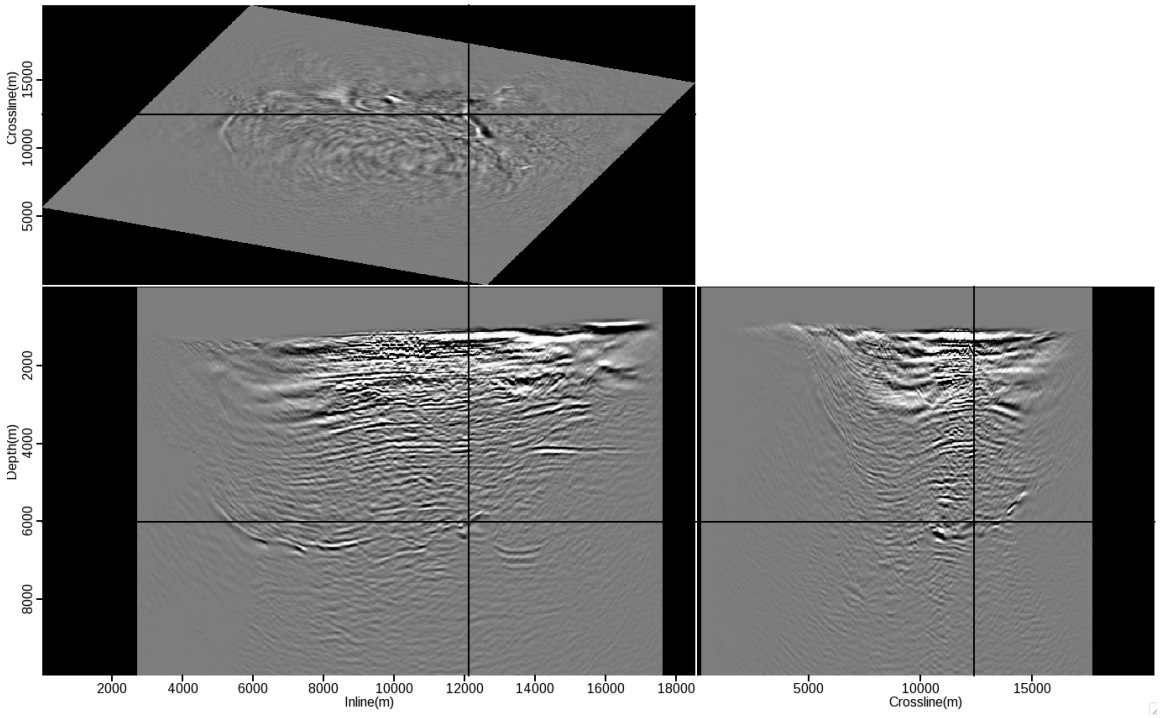

Figure 5.3: A rotated section of the RTM image, computed from the 103 OBNs. [**NR**] chap5/. fielddatartm

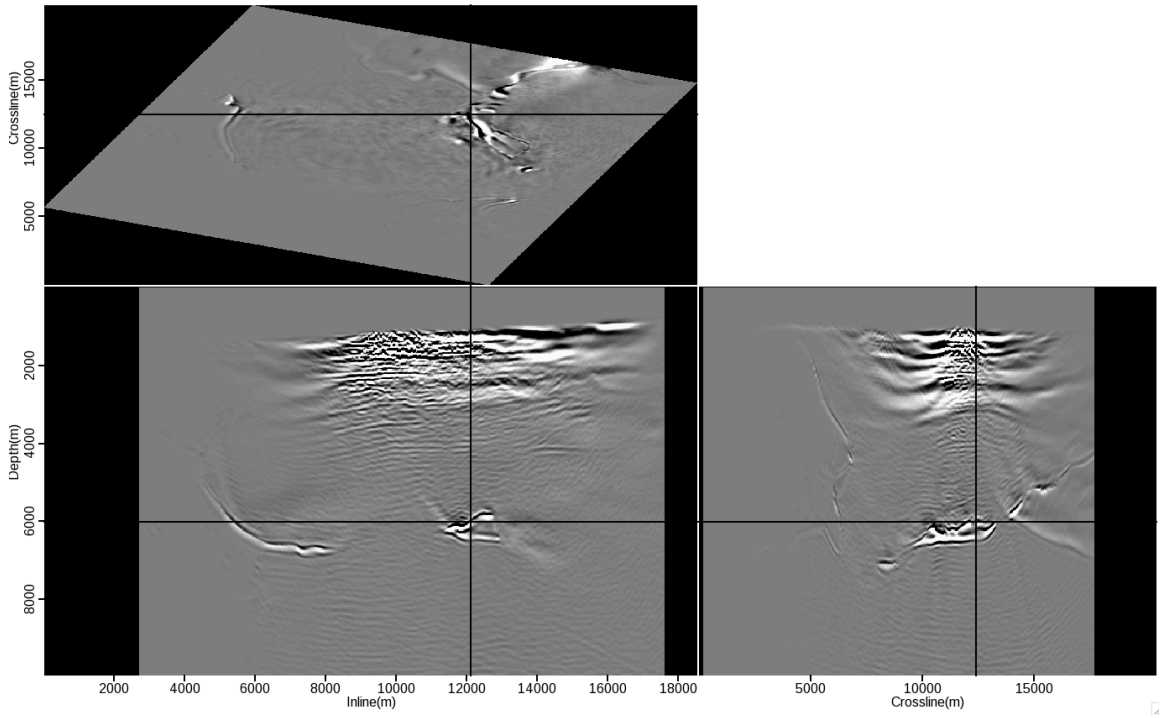

Figure 5.4: A 3D image at the same coordinates as Figure 5.3, after five iterations of LSRTM. [**NR**] chap5/. fielddatalsrtm

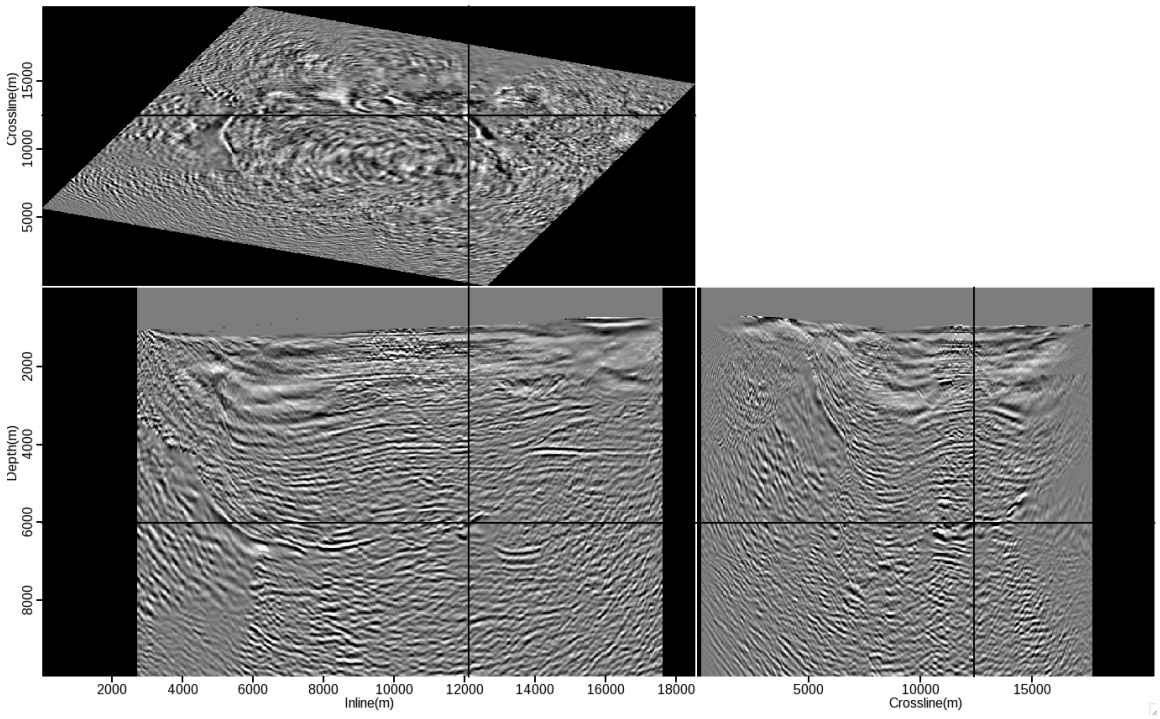

Figure 5.5: The same result as Figure 5.3, with AGC applied. [**NR**] chap5/. fielddatartmagc

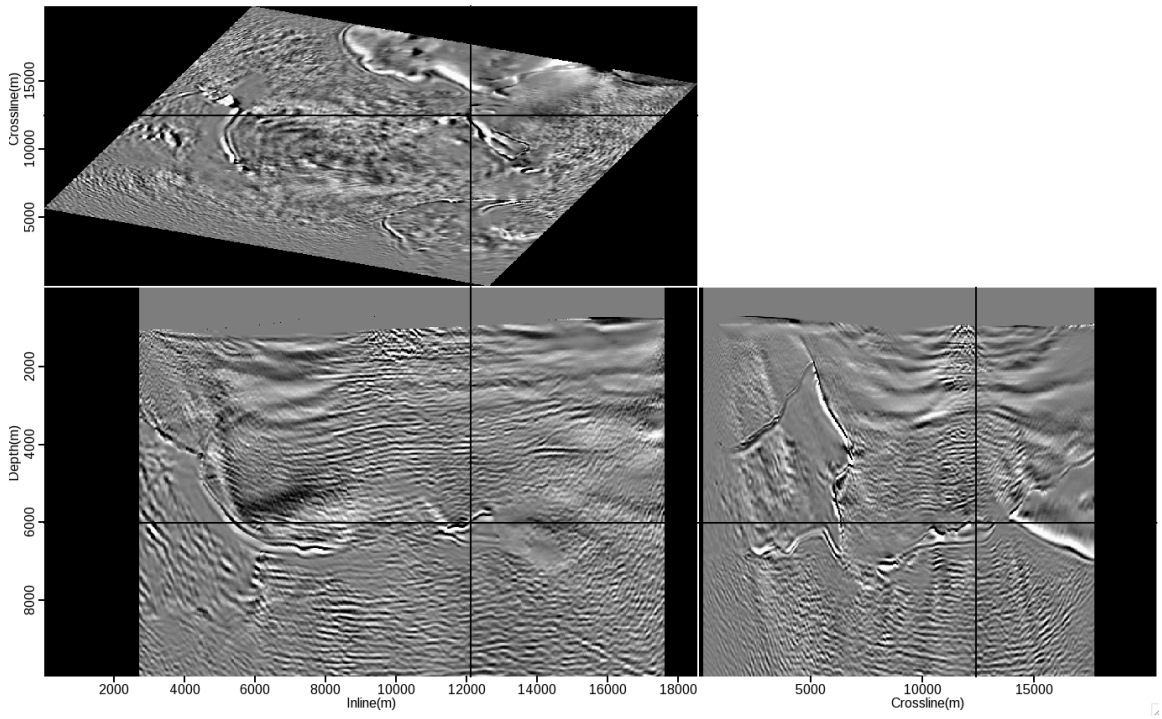

Figure 5.6: The same result as Figure 5.4, with AGC applied. [**NR**] chap5/. fielddatalsrtmagc

SOURCE SEPARATION

Simultaneous source shooting can be simply simulated with an OBN dataset, because the receivers are fixed and feature the same (or at least very similar) shooting pattern. Consequently, receiver gathers can be shifted and summed together, due to the linear superposition of these wavefields, to simulate blended acquisition. It is useful that we also have a conventionally imaged result, since we can image these data after separation and then provide a comparison between results.

The same 103 receivers migrated to create these images were combined into random subgroups of five shots, which were then delayed and summed together, with the delay times recorded. Figure 5.7 shows five such gathers combined, and then Figure 5.8 shows a windowed shot record from this continuous gathers. These subgroups were used to conserve some I/O cost (these receiver gathers are each 8 Gbytes in size), since reading very large time records can be arduous, but this does not invalidate the simulating of blended acquisition.

The same approach as Chapter 4 was used, which the operator Γ was used to appropriately window the shot of interest from the continuous recording, and image each of these separately. The separation approach used an extremely smoothed and slightly scaled version of the velocity model, and an isotropic, acoustic propagation kernel. Both of these attributes create a sufficiently inaccurate Earth description to test separation. The previous chapter simply smoothed and scaled the velocity, in this case the lack of additional TTI parameters will create further positioning errors. Separation was also performed using the correct velocity model, albeit with an isotropic, unextended separation engine. For both of these separations, ten iterations were used.

Extended imaging was applied in the inline direction, since this featured the densest sampling and the steepest dips. Twenty-five offsets were acquired in total, resulting in 625m of subsurface offsets imaged. This was done using a set of GPUs with a combined memory of about 52 Gbytes. For reasonable runtimes (same order of magnitude as TTI imaging), the entire image must be allocated on the set of GPUs,

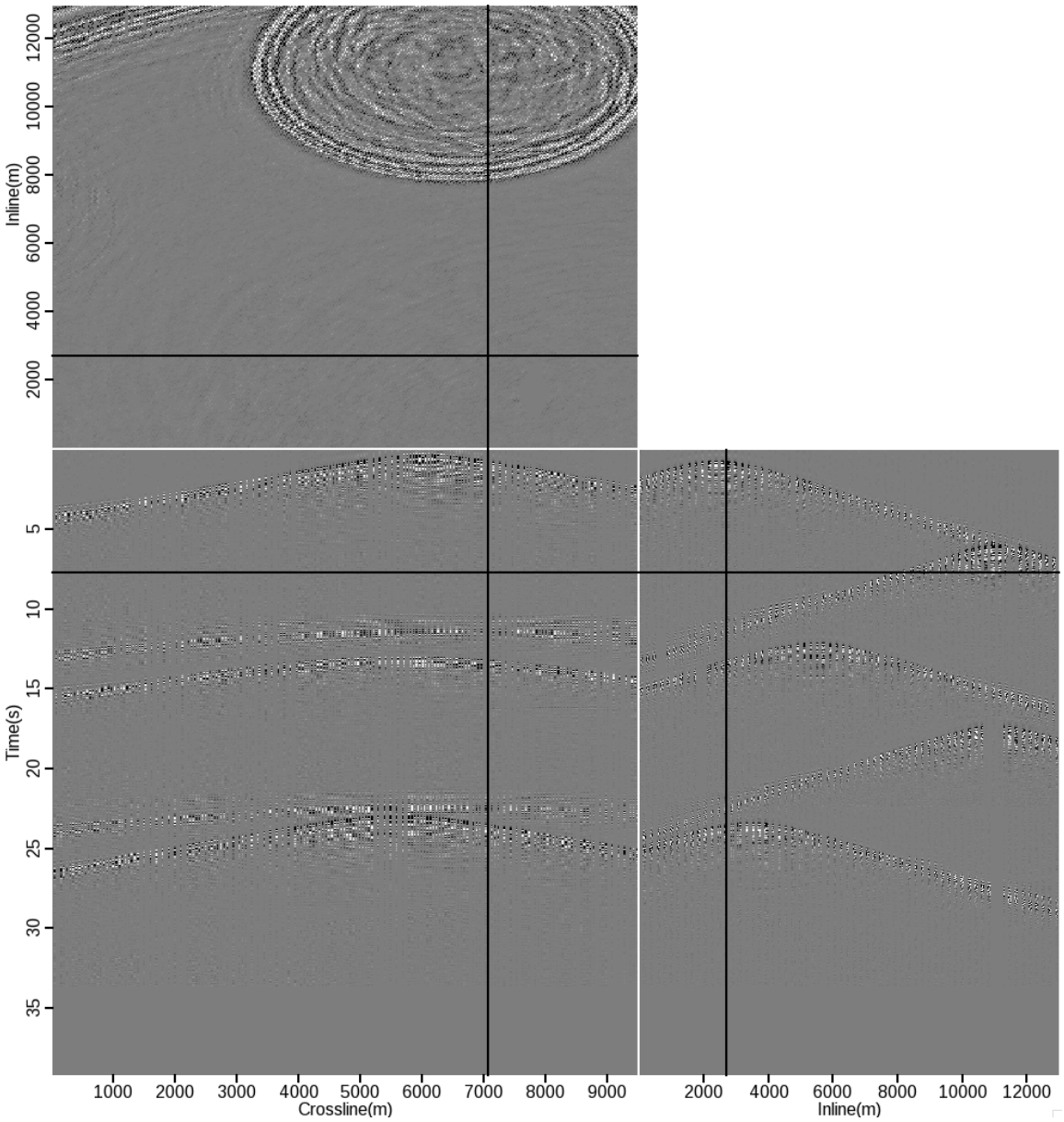

Figure 5.7: An example receiver gather after five OBNs are blended together. [**NR**] chap5/. cdata

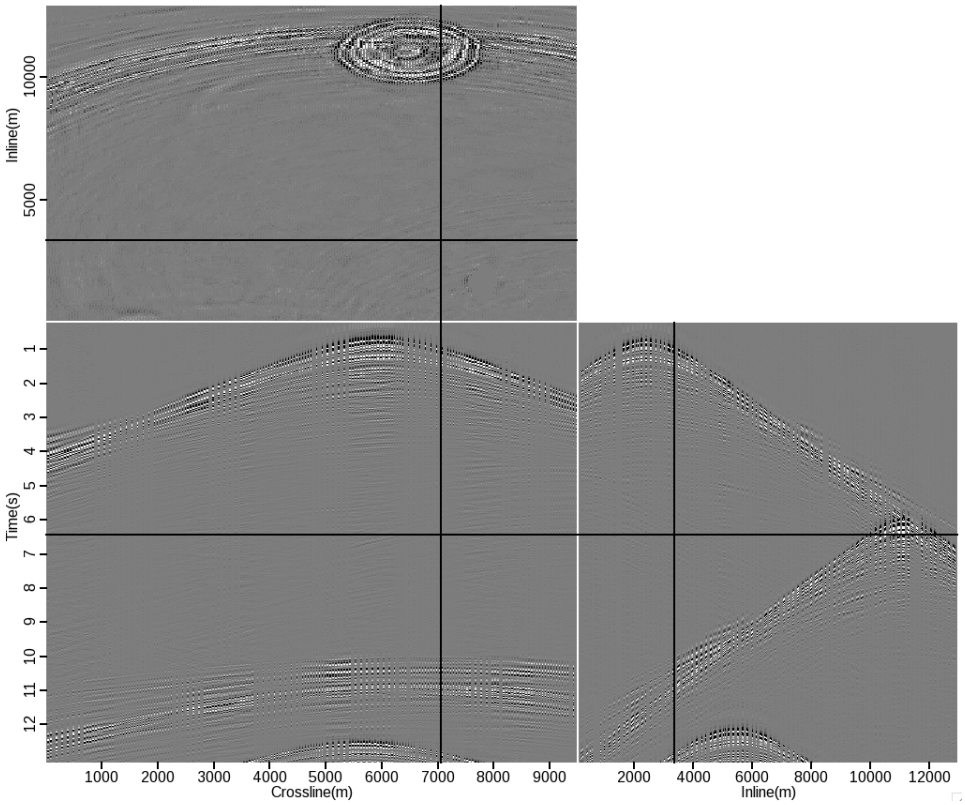

Figure 5.8: An example input receiver gather for the separation engine, which is a section of Figure 5.7 after Γ has been applied. [**NR**] chap5/. blendshot

as such a smaller area of the velocity model was used. The velocity was reduced to a cube of 400 Mbytes, selected by windowing around source positions, all 103 receivers were still used. Tests which offloaded the imaging to allow for more offsets to be acquired slowed the run time by a factor of thirty.

Figure 5.9 and Figure 5.10 show some inline image panels, windowed from the 4D image, with subsurface offset panels. Both image panels exhibit some imaging noise, since these are isotropic migrations of anisotropic data, and for the same reason neither panel is tightly focused at zero-subsurface-offset. However, Figure 5.10 is more loosely focused, and has energy spread over a wider range of subsurface offsets than Figure 5.9. It also shows some energy focused at non-zero subsurface offset. The zero-offset image from the correct velocity model result, and the full-offset image from the incorrect velocity model result, will now be used to attempt extended forward modeling.

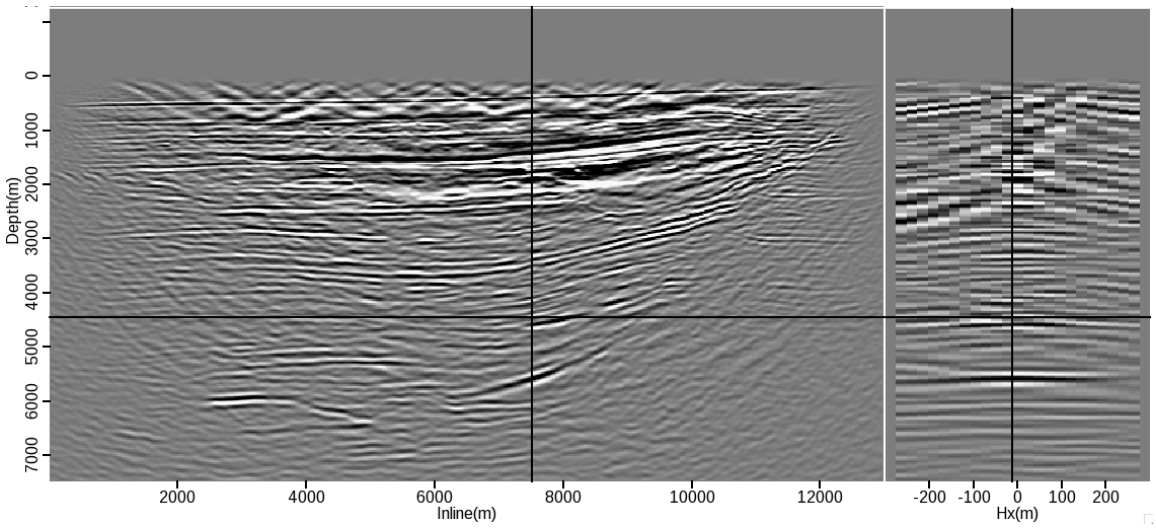

Figure 5.9: Isotropic, extended migration of these blended data using the correct velocity model. [**NR**] chap5/. cormodgather

An input receiver gather, a gather reconstructed with the correct velocity, and a gather reconstructed with the incorrect velocity, are shown in Figre 5.11; the location selected was in the middle of the domain. Whilst the kinematics are largely present in all three panels, there is some steeply dipping noise at large offsets present in the two

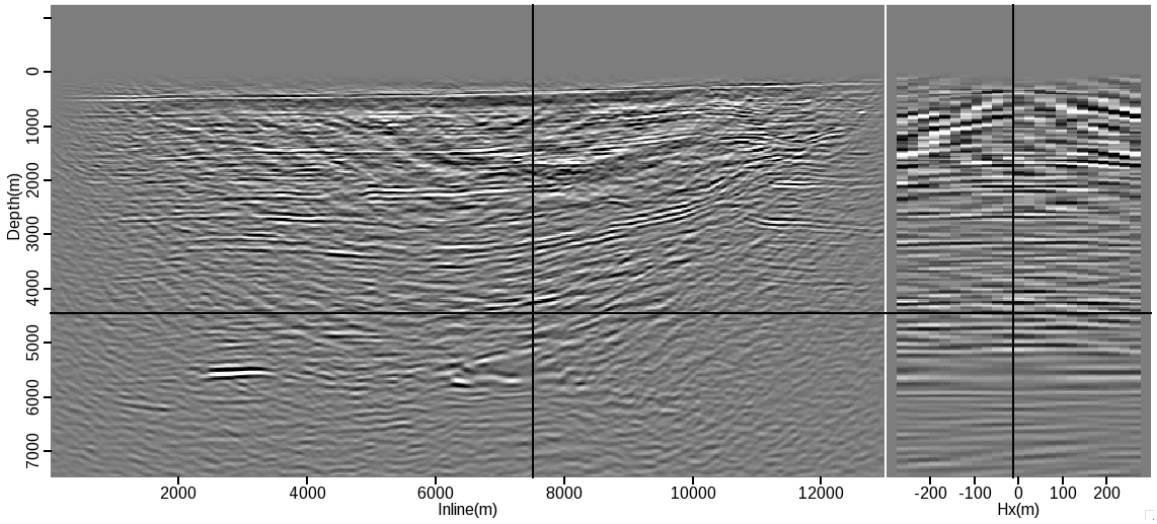

Figure 5.10: Isotropic, extended migration of these blended data using an incorrect velocity model. [**NR**] chap5/. incmodgather

reconstructed panels. Since this noise is present in both of the reconstructions, it is not due to extended Born modeling artifacts, which were observed in some of the synthetic tests. It is most likely due to these aforementioned image artifacts, induced from refractions. The salt bodies in these data often cause high velocity refractions, which can easily extend beyond reflection arrivals at large offsets. The limited crossline coverage, in terms of receiver density, means that some of this refraction noise will not stack out effectively in this dimension. This is an interesting phenomenon due to the geometry in question - blended energy stacks out easily, since it is random between shots, and randomly separated in time. Refraction noise, however, is harder to stack out without a variety of crossline receiver positions (these refractions also caused problems for LSRTM, above). Panels which include the recovered corssline data show this more clearly, and can be observed in Figure 5.12, Figure 5.13 and Figure **??**. Nonetheless, key reflection events can be mapped in all three panels, and the results of migrating these data show that the separation has preserved reflector information.

For a baseline comparison, these conventional data (unblended receiver gathers)

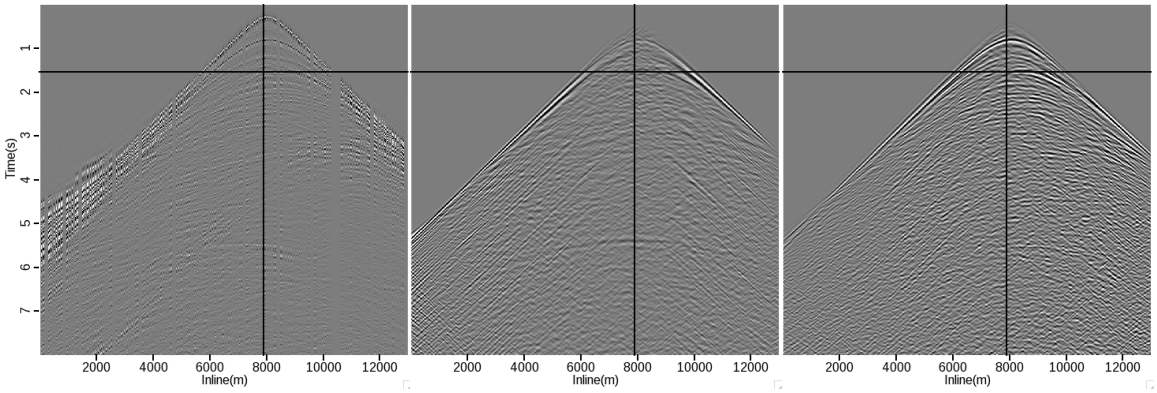

Figure 5.11: From left to right: an (unblended) receiver gather, the reconstructed gather using extended image space separation and an incorrect velocity model and the reconstructed gather using zero-offset, isotropic image space separation. [**NR**] chap5/. shotcomp

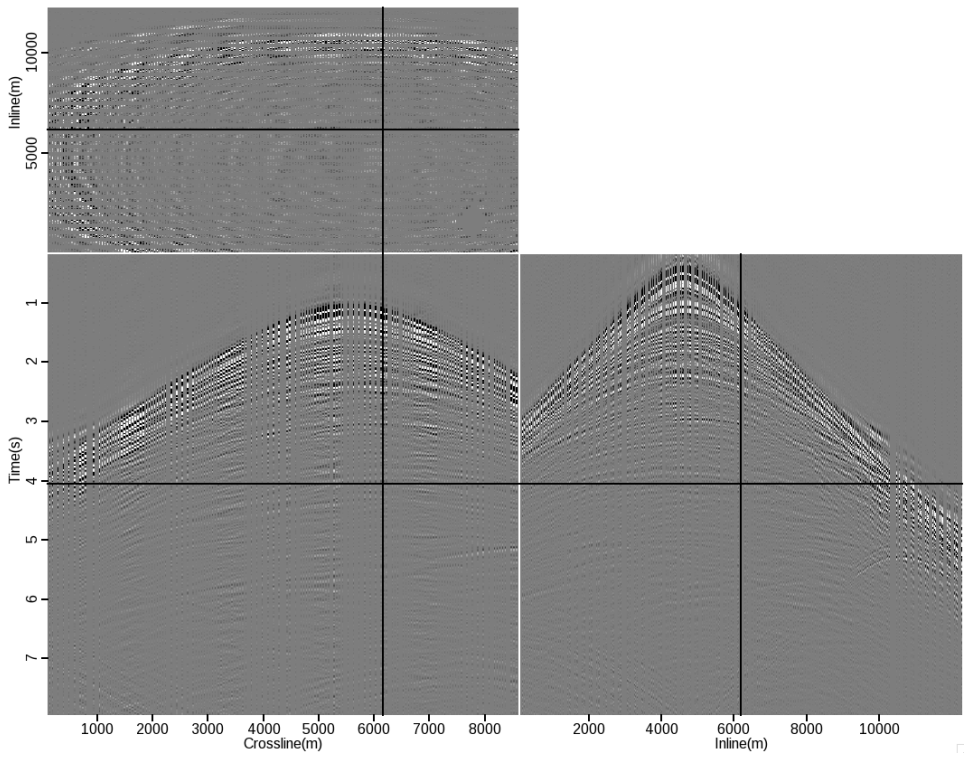

Figure 5.12: A raw, unblended receiver gather. [**NR**] chap5/. rcvclean

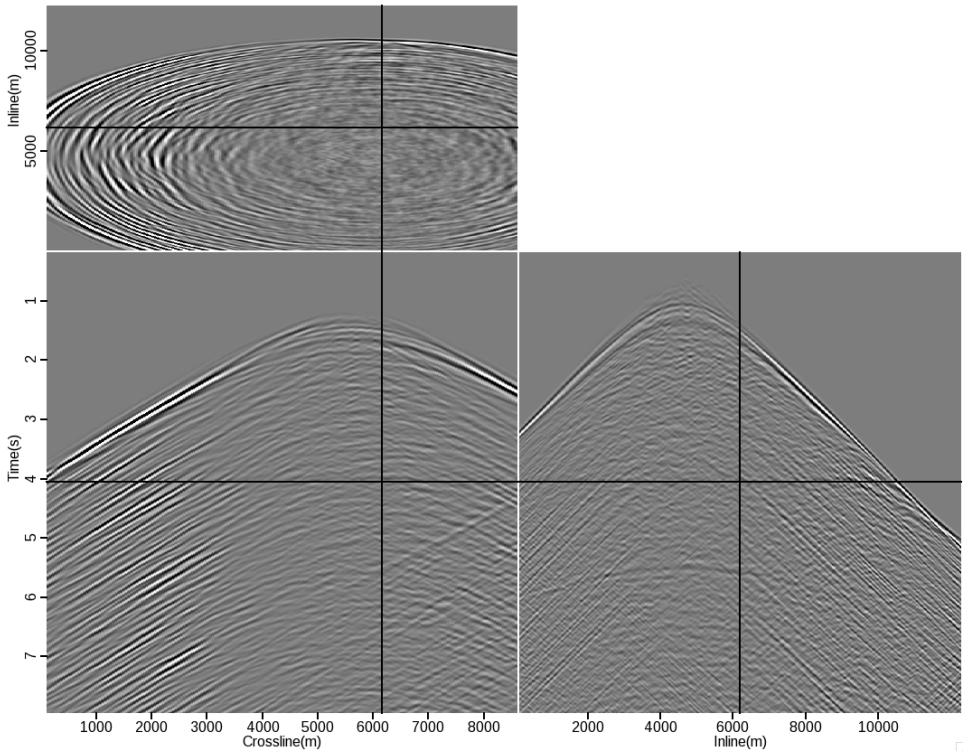

Figure 5.13: A receiver gather after extended unblending. [**NR**] chap5/. rcvinc

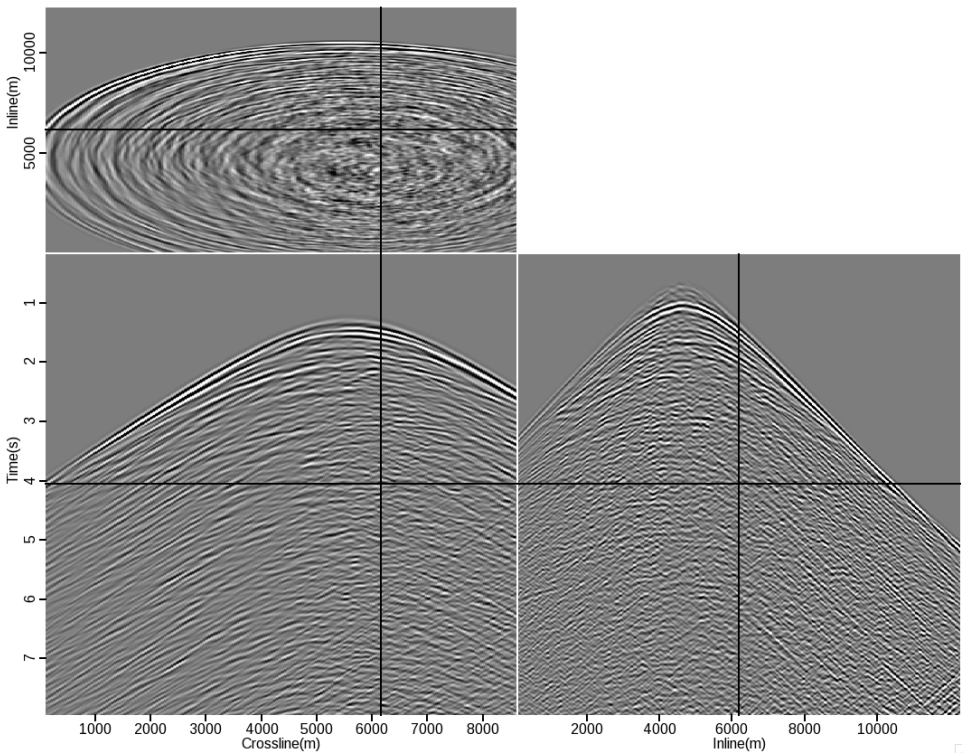

Figure 5.14: A receiver gather after unextended, isotropic unblending. [**NR**] chap5/. rcviso

were imaged over this smaller domain, and this baseline image can be seen in Figure 5.15. The image created using the data reconstructed using the accurate velocity can be seen in Figure 5.16, and the image created using the data with the incorrect velocity is shown in Figure 5.17. These are all TTI migrations of these respective datasets, illustrating the best possible image, given the input data.

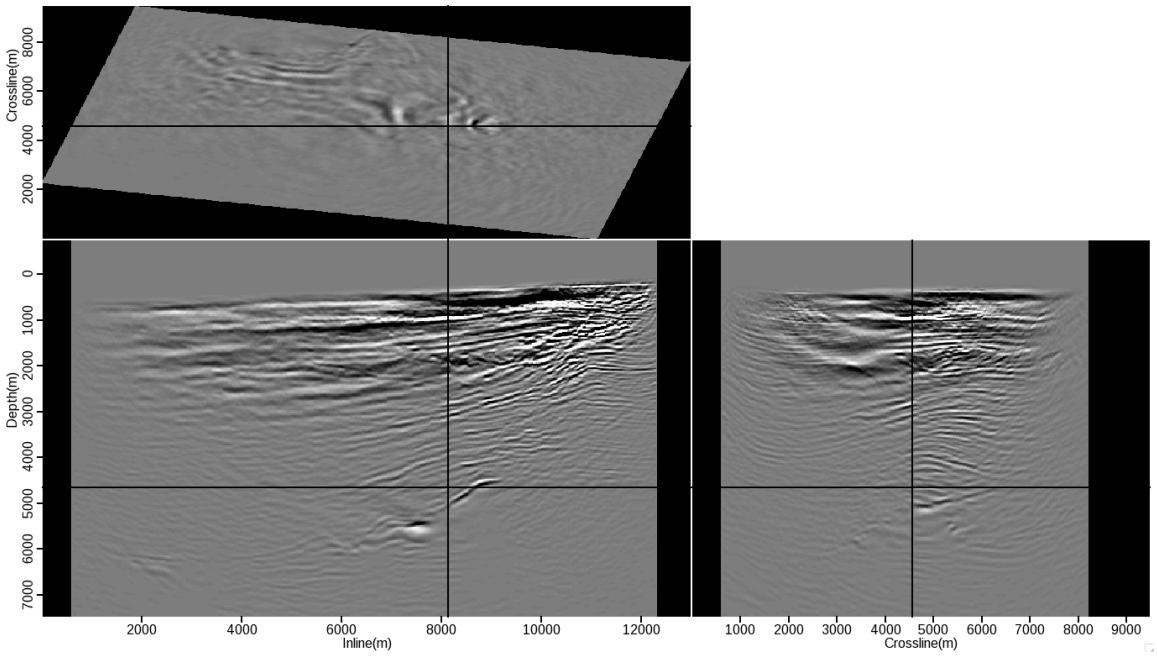

Figure 5.15: TTI migration of the input data before blending, for a baseline comparison. [**NR**] chap5/. baseline3dimage

These images are all directly comparable, and the vast majority of reflectors, contrasts and structures can be seen in all three. Expectedly, the wavenumber content in the reconstructed panels is a little lower, and the images obtained from the separated data are not quite as well resolved. An series of inlines, for ease of comparison, is shown in Figure 5.18. For unextended separation, it appears that some information around the salt bodies has been lost, from the subsurface offset panels shown above, this is not entirely surprising. Anisotropic parameters, especially ϕ, are often of higher value around salt bodies - thus, positioning errors are to be expected, and the lack of subsurface offset extension may cause some loss of signal in the image

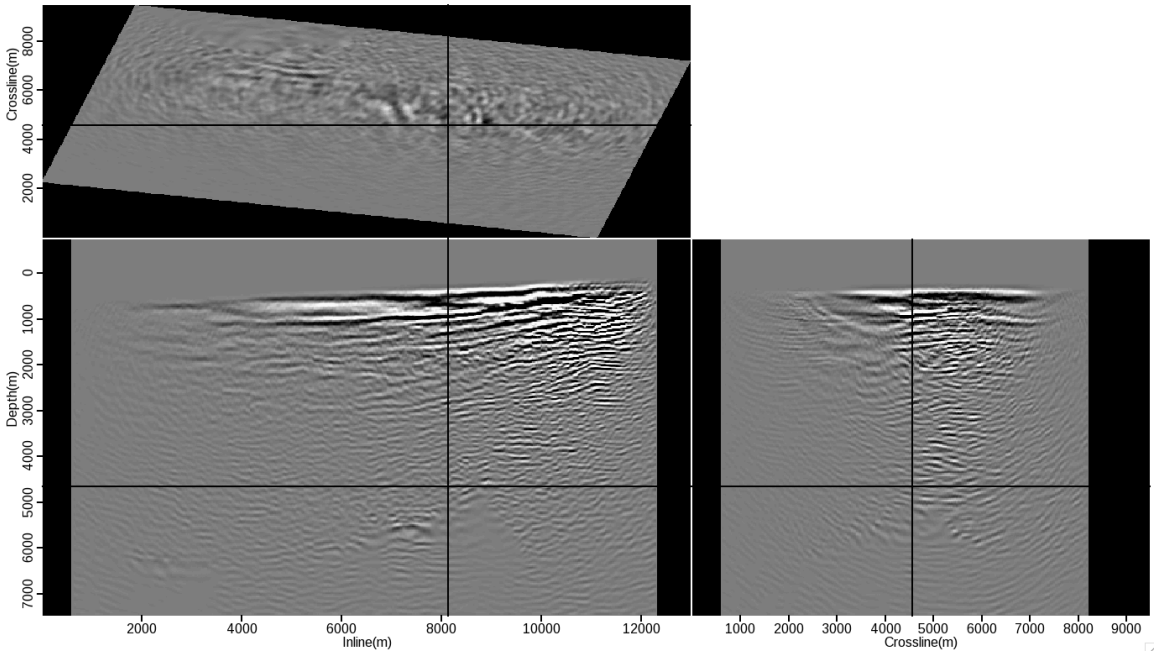

Figure 5.16: TTI migration of the reconstructed data after 10 iterations of zero-subsurface-offset, isotropic deblending. [**NR**] chap5/. zosep3dimage

space. These salt edges are better resolved in the image from extended deblending, however this image has a few other imperfections. There is some loss of continuity in the sediments above the salt, and the wavenumber content is lower.

Some of this noise, and continuity disruption, stems from an implementation limitation - only one axis was extended during initial migration. For this dataset, the choice of axis was obvious, since the source sampling along the inline direction was much more consistent and dense. Despite the limited dip content of the crossline direction, it is probable some information was lost at this point. By extending the migration in both x and y, energy would not be lost in this manner. Another choice would be to use time lag imaging, as this would capture unfocused arrivals in any spatial direction. However, the restraints of the computers and the input data size meant that multiple spatial lags, or time lags, would be close to unfeasible.

Despite only extending one axis, it is encouraging that dipping reflectors at the

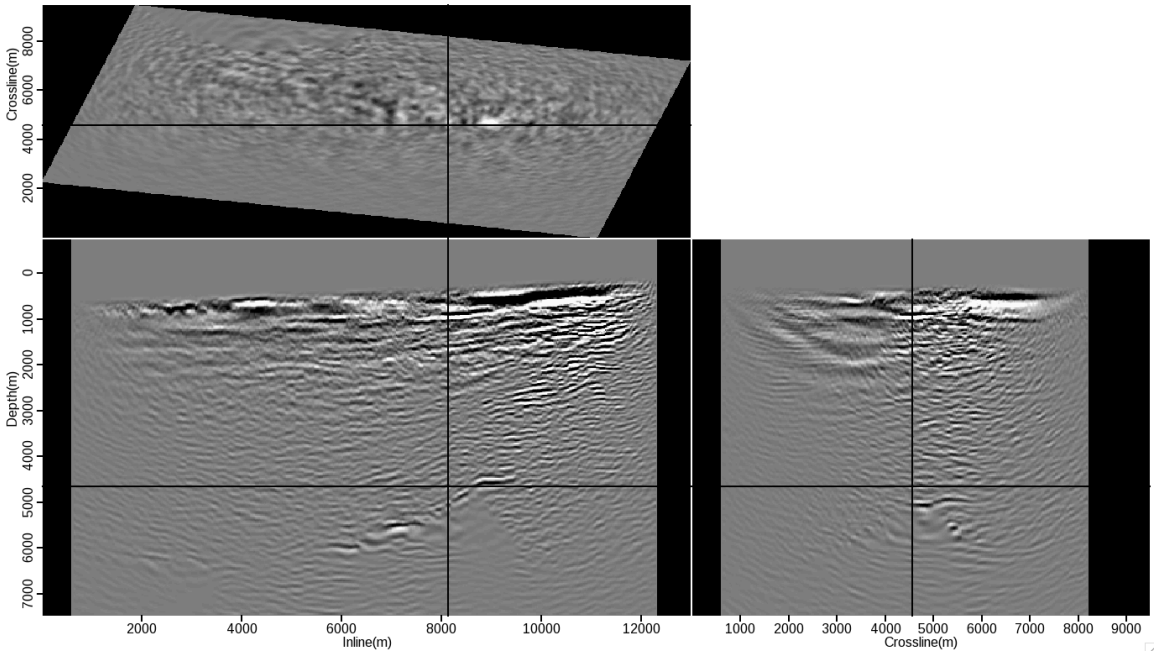

Figure 5.17: TTI migration of the reconcstructed data after 10 iterations of extended, istropic deblending with a rough velocity model. [**NR**] chap5/. extsep3dimage

edge of the crossline panels are still imaged in all three cases. This helps to confirm that only marginal information was lost in the extended deblending, and further iterations would improve the results.

As discussed in the previous chapter, image-space separation can be augmented into a model-building and imaging process. In this context, then these results are even more encouraging. All output images well represent the geology, structure and amplitude character of the subsurface, and as the model improves, then this representation will become stronger. Furthermore, all would act as powerful starting models for any sort of subsequent linearized inversion. If this scheme were to be combined with, say, migration velocity analysis, then an effective positive feedback loop could be created. The extended images are already created, which can be used to provide a velocity model update. This updated model could then be used to improve the separation fidelity, and the process repeated.

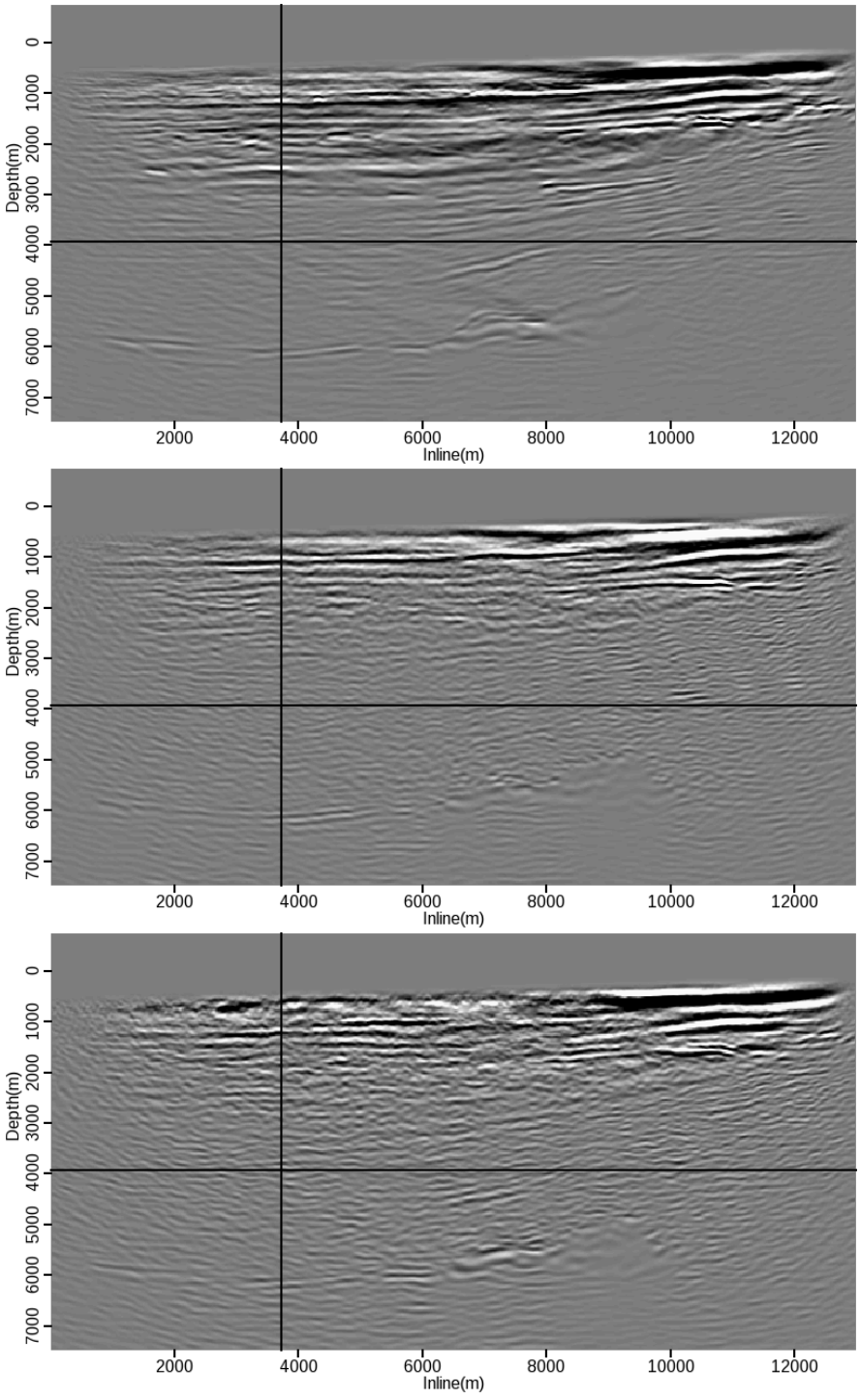

Figure 5.18: An inline comparison of the images shown in Figure 5.15, Figure 5.16 and Figure 5.17. [**NR**] chap5/. inlinecomp

CONCLUSIONS

These results demonstrate that extended image space blended inversion is effective for shot separation. Given a blended input dataset, this technique was effective in separating these shots into separated gathers using both an accurate and an inaccurate velocity model. Furthermore, the separation engine was isotropic, and the velocity model was built under the assumption of TTI anisotropy, so both attempts will exhibit inaccuracies in the migration.

Imaging these separated datasets was successful and representative, however, both results suffered from a reduction in frequency and a variety of implementation artifacts.

ACKNOWLEDGMENTS

Special thanks to Scott Morton, of Hess Corporation, for making all the work in this chapter possible. Scott helped with data access, concepts, interpretation and practical implementation discussions. Special thanks also to Wayne Lee and Faqi Lui, of Hess Corporation, for their assistance in harnessing the computational system, and analysing results, respectively.

Chapter 6

High performance computing solutions

IMPLEMENTING LINEARIZED INVERSION

As discussed in Chapter 2, and indeed throughout this thesis, practical implementation of these wave-equation based inversion schemes is not straightforward. Naively coding such a system would result in impractically long run times; for any significantly sized problem large computers much be used and care must be taken to use them to the best of their ability.

This final section will demonstrate how these operators can be optimally coded for both a system of CPUs and a system of nodes that contain both CPUs and Graphical Processing Units (GPUs).

Optimization

As a concept, optimization is an act or process that aims to make something as effective as possible. In applied mathematics, optimization is posed as a way to maximize productivity by finding the values and influences of controllable factors which

determine the behavior of a system. This chapter will address computational, or programming, optimization. There are two major requirements to consider computational optimization: intimate knowledge of the computer (or network) that one is optimizing for, and intimate knowledge of the serial and parallel sections of the algorithm. The 'controllable factors' for this variety of optimization are the hardware and it's acceleration options (multi-core, vectorization etc), the algorithm (decomposition, parallel segments etc) and time.

For research code, time is a crucial factor for computational optimization - it is imperative that the additional time required to optimize the code, t_{opt}, is significantly less than the difference between the new runtime and the unoptimized runtime, say $t_{opt} \ll t_{orig} - t_{new}$. Put into words: time saved through optimization must be less than the time spent to optimise. For production code, that will be run many times, then optimization can save vast quantities of future compute time.

Furthermore, the abstract conception of optimization implies a final solution. When it comes to computational considerations 100% optimization is never achieved, as one could never balance $t_{opt} \ll t_{orig} - t_{new}$. From hereon the term optimization will denote an effort to accelerate computational runtime by any increment. This will be in particular reference to two systems, multi-core CPUs and linked GPUs. The focus will be primarily on the latter, nonetheless CPU techniques will be discussed in brief in order to construct a valid comparison.

CPU based linearized inversion

For optimisation, it is imperative to understand the architecture of the given card(s). Figure 6.1 shows a simplified diagram of the separate memory levels for a typical four-core CPU node (modern nodes may contain eight or sixteen cores, this is for illustrative purposes.) Each core features an FPU (Floating Point Unit), which performs the arithmetic, three individual memory levels (local to each FPU and only accessible from that FPU) and two larger memory levels, accessible to all FPUs on the node. As 'distance' from the FPU increases so does both size and latency of the

memory level; latency is a measure of the time taken to transfer a byte from a given memory location to the FPU.

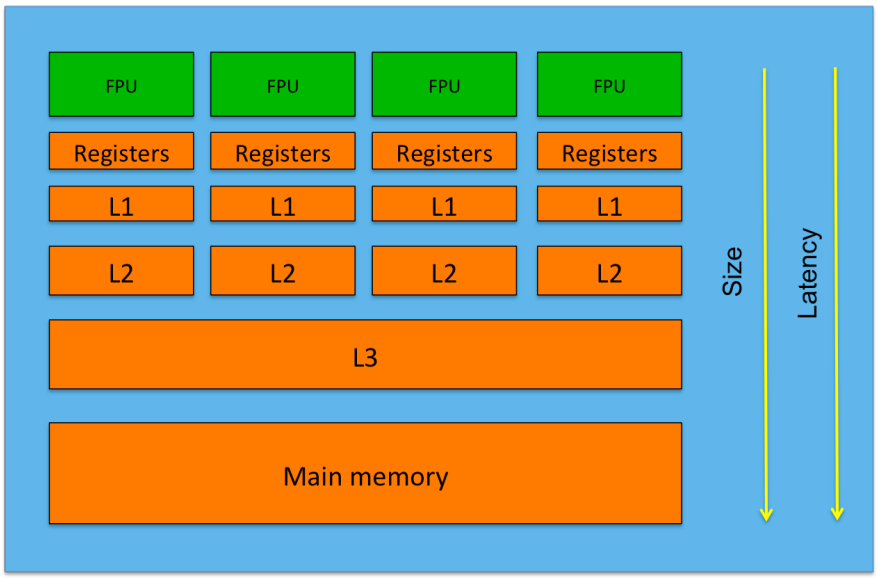

Figure 6.1: Diagram of the memory layout of a typical multi-core CPU node. As distance from the Floating Point Unit (FPU) increases, so does both the size and latency of the memory unit. [**NR**] chap6/. multicpu

From this model there are already two options to consider - the fact there are multiple arithmetic units, and the fact there are memory levels of different latencies. Initially, making use of the additional FPUs will be addressed. There are many programming languages that can take advantage of multiple CPU cores on a single node - MPI, POSIX, Python etc, however this discussion will be limited to OpenMP (Chapman et al., 2008). OpenMP is an advantageous choice due to the ease of its initial implementation; only a few additional directives are required in order to make a code run over all available cores. For high performance optimization this is also it's disadvantage, since these compute threads are not strictly independent and speed up saturates. However, the goal of this discussion is to provide a semi-optimized CPU code for reasonable GPU comparison, hence OpenMP plus some additional optimization is acceptable. Of course, when it comes to parallel programming one is fundamentally limited by the serial sections of the process, and speed up saturation is

also seen due to this. This phenomenon is known as Amdahl's Law (Amdahl, 1967).

Modern CPU nodes allow multi-threading, whereby a given core can run multiple threads. For some applications this can be an advantage, but due to the size of the fields allocated in seismic wave propagation and the relative operation-to-read count more often than not multi-threading results in a performance decrease. Due to this only one thread per core will be used, so during CPU discussions these terms can be considered interchangeable. OpenMP directives were added to the outer (slowest) loop of the propagation kernel (Chapman et al., 2008). The node used comprised of an Intel Nehalem CPU running eight independent cores, tests were run on one, two, four and eight cores. The results of improved run times can be seen in Figure 6.2.

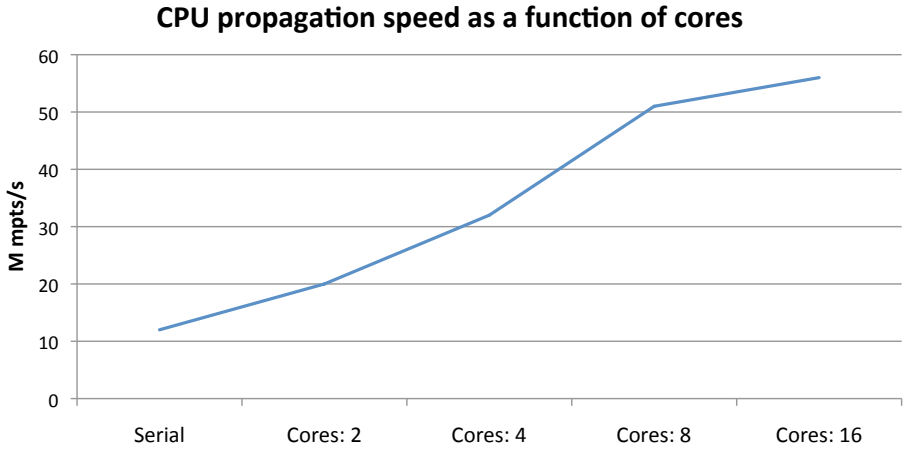

Figure 6.2: Compute speed (million models pts calculated per second) against implementation. [**NR**] chap6/. cpuspeeds

Noticeably, run times improved. However we see the relative speed up as a function of number-of-cores saturate very quickly; this is due to a number of reasons specific to CPU architecture. These explanations revolve around moving between shared memory spaces and the context switching necessary to do this. While the latter of these issues is unavoidable, the former can be addressed by further considering the memory levels shown in Figure 6.1.

To make full use of these low latency (cached) memories it is desirable to fill

them with information that will either be re-used in the algorithm, or fill them with values that are aligned for computation (contiguous). This is called temporal and spatial locality (Denning, 2005). Spatial locality refers to values close to each other in physical memory, particularly those that may be read in a cache line; temporal locality refers to values that have been read into a cached memory already and can be efficiently reused before they are flushed.

For wave propagation, values that could be reused are local wavefield values (used for the derivative) and local velocity values (reused during adjoint propagation). When convolving with the finite-difference stencil run-times will be much faster if the information needed resides on L1 or L2 memory. Once the code reaches for values in L3 or global memory then run times can increase dramatically. Whilst L1, L2 and register memory feature very low latency they are both very limited in size and for generic (non-machine specific) coding there is no explicit control over how the memories are used. It is possible to encourage the CPU to fill and re-use the cache levels in certain ways however it is very easy to pollute these cached memories.

A typical velocity field is of the order of 1000x1000x1000 model points in the three Cartesian directions, stored in 4-bit precision, giving a total size of 4 Gbytes. This is clearly vastly larger than even L3 memory so the problem must be divided into smaller blocks, this is called 'blocking' which is a type of 'domain decomposition.' The stencil used is a symmetric 8th order finite difference stencil, making it 25 pts (in 3D).

Initial estimates of how large to construct these blocks is to ensure they fit into L2 memory, thus individual FPUs do not have to access shared memory spaces in order to perform the key computations. Secondly, the shape of the blocks must be considered. The stencil is applied by looping over the three principle axes, with the loop over the inner axis being the 'fast' axis (the axis along which memory is contiguous.) This is the only axis for which it is possible to be confident over efficient memory reuse (spatial locality), so the most efficient cache coherency is observed by constructing long, thing, 'pencil' shaped blocks. In this working example the fastest axis is the depth (z) axis; by making the length of this axis substantially longer than the other

two (x and y) then a speed up of around double is observed, purely through blocking.

Doubling the compute speed is a promising result, however it is far below the maximum possible speed up. A new plot detailing how blocking can assist in propagation speeds can be seen in Figure 6.3. It would be possible to further optimize the block size and the way the third axis is looped over; it would also be possible to extend the optimization attempt to take advantage of the vector units contained within the CPU cores. However the next step within this thesis will be to consider how GPU computing can assist the acceleration of acoustic wave propagation. The attempt described herein for CPU optimization creates a valid comparison to GPU run times, since both multi-threading and memory caches have been used to improve run times.

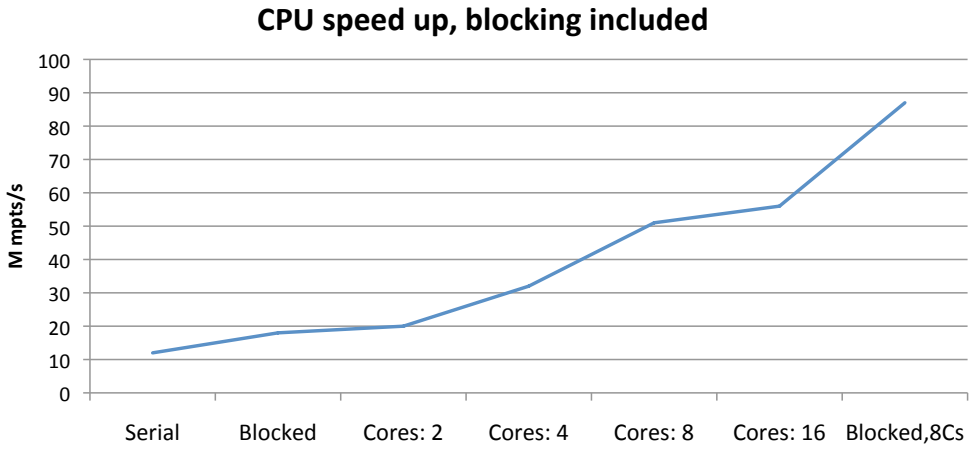

Figure 6.3: Compute speed (million models pts calculated per second) against implementation. [**NR**] chap6/. blockspeeds

GPU BASED LINEARIZED INVERSION

Graphical Processing Units (GPUs) have been present inside computers for decades. They feature a highly parallel specialized electronic circuit which traditionally is used to output graphics to a display. The inception of General Purpose GPUs (GPGPUs) in the early 2000s allowed practical computing on these devices by virtue of

manipulatable floating point support (Fung et al., 2002), however it wasn't until the advent of CUDA (Compute Unified Device Architecture), by NVIDIA in 2007, that the potential of this new programming paradigm could be easily harnessed by general developers (Manavski and Valle, 2008).

Many of the world's largest supercomputers now feature CPU-GPU hybrid capability. CUDA is very similar to C, but also allows GPU specific routines (kernels) to be written and run. Generally the CPU is referred to as the 'host,' and the GPU as the 'device.' For efficient GPU implementation an algorithm must feature two key characteristics: it must be scalable and parallelizable. The former of these simply means that as the problem size is reduced or increased the relative performance should be roughly preserved; this requirement is needed because the GPU primarily relies on fine-grain parallelization. The latter means that the problem must be able to broken into smaller sections that can operate, at least for a period, independently. Any algorithm that fits these criteria should be accelerated via GPU computation ((Ohmer et al., 2005); (Foltinek et al., 2009)). Certain methodologies, such as some classes of triangulation, can be slowed down when ported to a GPU, but the vast majority of algorithms can be adapted to harness the parallel power of GPU computing.

Figure 6.4 demonstrates the layout for a simplified GPU, and this can be compared to Figure 6.1, which shows the layout for a simplified multi-core CPU (the corollary, in this case). The GPU layout features many more Streaming Multi-Processors (SMPs) than that CPU; whilst a typical CPU will have around 8 cores (without multi-threading), a given GPU can have upwards of 1536. The exact cards used for the majority of this study were the Nvidia Fermi M2090 cards, which each feature 512 SMPs. However the true parallel power of the GPU is not exhibited solely due to these increased numbers, but by the fact that these individual SMPs (or thread-blocks) can operate 100% independently, without the need for context switching. Whereas the individual FPUs on a CPU node can operate simultaneously, the layout requires constant communication between memory levels and FPUs, which (especially for a naive implementation) severely inhibits performance (Mallón et al., 2009).

Furthermore, each of these thread-blocks is divided into a two-dimensional grid

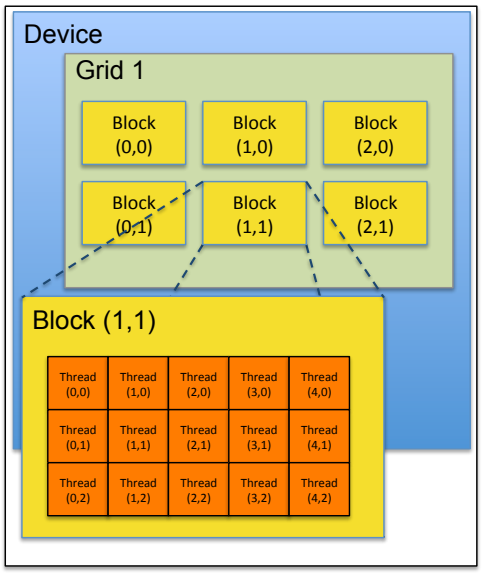

Figure 6.4: Schematic for a given GPU architecture, demonstrating how the grid is broken into two-dimensional thread blocks, which are broken up into individual threads. [**NR**] chap6/. GPUarch

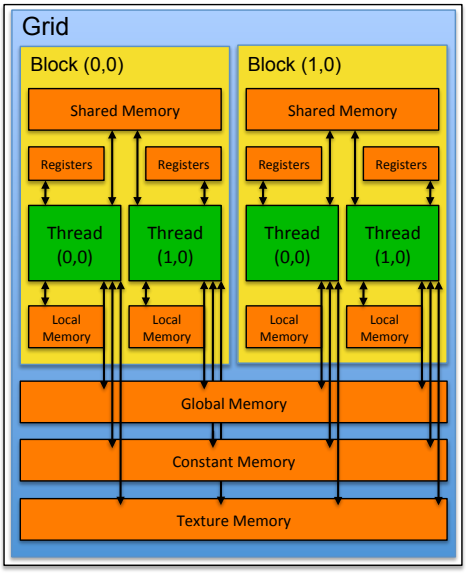

Figure 6.5: A visual representation for the different memory levels within a GPU. Each block has a unique shared memory and each thread has it's individual local memory and set of registers. Global, constant and texture memories are shared between all thread blocks. [**NR**] chap6/. GPUmem

of threads (figure 6.4). These 'threads' are the individual processors that perform the work; similarly to the thread-blocks, many threads can run independently and simultaneously. The caveat is that each thread must be performing exactly the same task (both within and across SMPs) on a reduced portion of the problem - hence, the algorithm must be scalable and parallelizable. By executing thousands of these individual threads simultaneously all latency is effectively hidden, and very high levels of multiple parallelism are possible. This latency hiding through mass thread execution is what allows the GPU to operate at such a high bandwidth.

The different memory levels of the GPU are shown in figure 6.5, again this can be compared to figure 6.1. Each thread-block features an individual and unique shared memory level, values stored here can be different between thread-blocks. This is analogous to the L1 cache in the CPU - only around 16kb or 32kb in size, and very low latency. Additionally, each thread has a separate local memory and set of registers, but these have to same for every thread across the whole GPU. The memory levels of significant size are global, texture, and constant memory, and these all share the 6 Gbytes of DRAM on the card (for the M2090, other vintages vary in DRAM size). These are similar to the global memory on the CPU; they are shared between all SMPs and have a much larger latency. In fact the latency for a thread to access data from the GPU global memory can be up to four times as large as that for a given FPU to access data in the CPU global memory. Thus for the GPU to out-perform the CPU intelligent use of shared memory is imperative, in addition to concurrent thread execution.

Single card GPU optimization

The global memory on a single GPU card is drastically smaller than a CPU. A Fermi M2090 features 6 Gbytes of addressable RAM, whereas a typical CPU node might feature 16x, 32x, or even more times this. Concordantly, a solution that uses multiple GPUs must be used, such that large 3D problems can be solved. However, developing an optimized single card solution for a modest problem size is the first step for both

code testing and strong scaling performance evaluation.

To maximize single card performance it is important to understand how groups of threads are executed. When a kernel is written and called it gives each thread the same set of instructions to execute, and understanding which memory points these threads access is crucial. During execution threads are grouped into warps, which are simply a set of 32 aligned threads. A single warp is able to access 32 bytes (a GPU cache line), essentially for free. Any thread within this warp can access any of these bytes with negligible latency, however if a thread tries to access an address that belongs to a different warp then context switching must be used, which briefly stalls the bus. Figure 6.6 demonstrates this pictorially. Initially each thread is accessing a single memory point within it's warp, and bus utilization is 100%. Next, each thread is operating on scattered memory points within the warp, but because it does not reach outside the given 32 bytes bus utilization is still 100%. In the third case the address that the threads are instructed to access do not all lie within the correct warp and there is a systematic shift, this creates a stall and utilization is reduced to 80%.

Fortunately, for finite-difference type operations warp alignment can be often be achieved by simple padding. In the case of applying an 8th order finite difference stencil out-of-warp access occurs due to the half-stencil width (halo) reaching outside of the warp, the case shown in figure 6.6. Padding the computational domain, at the beginning, by 28 points instigated a performance increase of almost 20% because this systematic shift was negated.

Whilst warp alignment is important for maximizing GPU performance potential, ensuring that the memory levels are used optimally is far more crucial. If, for example, an RTM implementation does not make use of shared memory then the GPU exhibits a worse performance than an eight core CPU node. This is entirely due to the aforementioned high latency of GPU global memory, despite the high level of parallelism. The implementation must aim to minimize any global memory access during thread execution. For the propagator the difference between CPU and GPU compute speeds can be seen in Figure 6.11. Note that the most naive GPU methodology, which does not use shared memory, is slower than the 8-core, blocked CPU

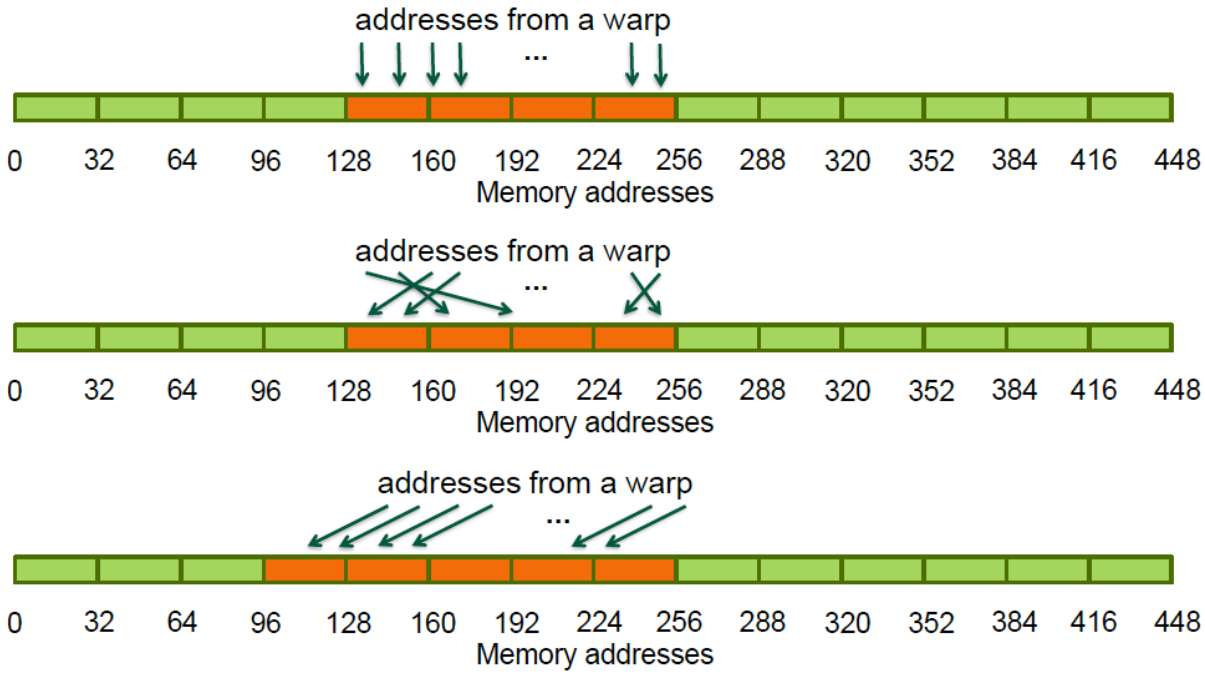

Figure 6.6: Diagram of warp alignment with memory requests. The above two cases feature 100% bus utilization, the bottom case has 80% due to misaligned memory accesses. [**NR**] chap6/. warps

code.

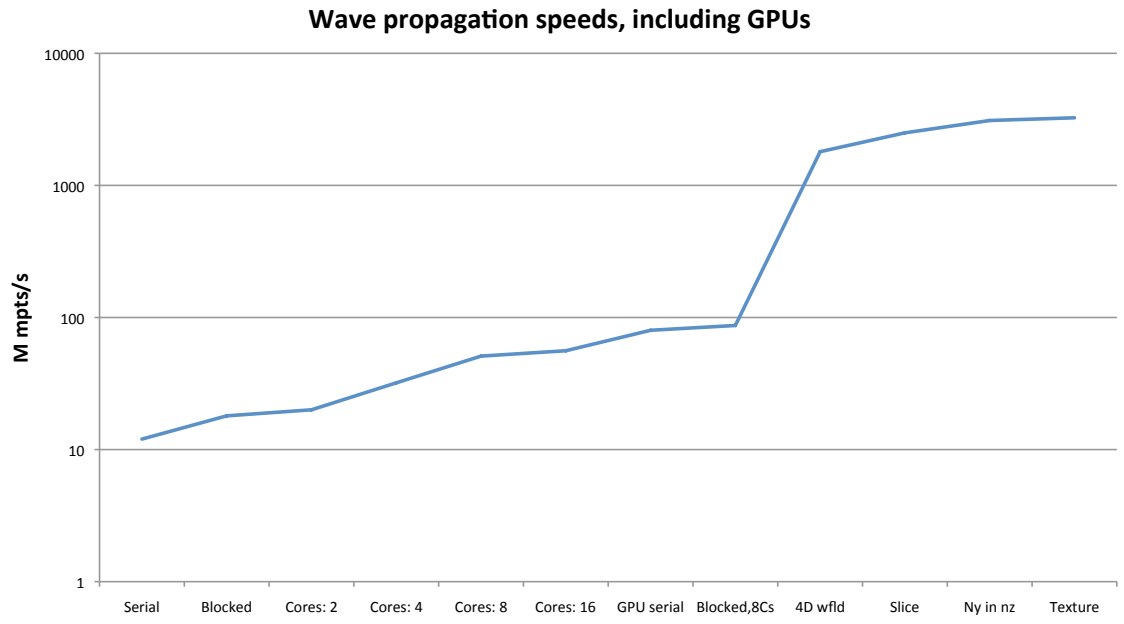

Figure 6.7: Compute speed (million models pts calculated per second) against implementation. [**NR**] chap6/. gpuspeeds

Random boundaries

A major bottleneck in RTM stems from the complex conjugate in equation 2.5. The consequence of this is that the sense of time for the source wavefield is opposite to the receiver (data) wavefield, thus one of these must either be stored or propagated twice (Baysal et al., 1983).

There have been multiple solutions to this source wavefield problem, such as source wavefield saving, checkpointing (Symes, 2007), compressive sensing and random boundaries (Clapp, 2008). The most intuitive solution is the first of these - to simply save the entire source wavefield and read back the correct sections as necessary. This will often cause an I/O bottleneck as disk access can be very slow. This methodology when using a GPU is disadvantageous; the GPU can not read from the

disk so all disk access must be first routed through the CPU. This creates even more of a bottleneck, especially since now propagation is accelerated, disk access appears as a larger time fraction. On the system used for these tests, disk access stagnates at around 200 Mbytes/second. Checkpointing is a way of favoring extra computation over disk access, and under certain circumstances can provide advantages for CPU based RTM. However it requires multiples source wavefield slices to be held in memory concurrently, and the global memory of the GPU cannot support this.

Random boundary based RTM favors extra computation over disk access; all disk based wavefield access can be mitigated. Generally, absorbing domain boundaries are used, this is to avoid high-amplitude, coherent and non-physical boundary reflections. Simply put, any wavefield incident on the computational boundary is artificially removed before it can reflect. These conditions work well for artifact reduction, however the conservation of energy is violated and this propagation is not time reversible. If source propagation was time reversible then disk access during the adjoint process could be avoided. It is possible to instead pad the domain with an increasingly random velocity field. An incident wave will scatter incoherently as it approaches the boundary, and no energy is artificially removed.

Thus, by randomly padding the domain, propagating the wavefield to the final two time slices and saving these, it is possible to reverse the sense of time and exactly recover the initial wavefield (to within machine precision.) Practically, for RTM the source wavefield can first be propagated and the final slices saved, then when back propagating the receiver the source can be back propagated concurrently, and the imaging condition applied at each appropriate time step. This is summarized in algorithms 7 and 8.

Of course, this technique introduces significant quantities of random noise into the system. Fortunately, this noise is random and incoherent, and will stack out between time steps and between stacked shots. This noise reduction follows a random walk type reduction, and experience shows that by using ten seismic sources or more that this random noise has already been reduced to background noise levels.

Algorithm 7 RTM with source reading

```
while it < nt; it++ do
  Propagate source wavefield
  if imaging time step then
    Save source wavefield slice to disk
  end if
end while
while it > 0; it– do
  Propagate receiver wavefield
  if at imaging time step then
    Read source wavefield slice
    Correlate source and receiver
  end if
end while
```

Algorithm 8 RTM with random boundaries

```
while it < nt; it++ do
  Propagate source wavefield
  if it=nt or it=nt-1 then
    Save source wavefield slice to disk
  end if
end while
while it > 0; it– do
  if it=0 then
    Read source wavefield slices
  end if
  Propagate receiver wavefield
  Propagate source wavefield
  if at imaging time step then
    Correlate source and receiver
  end if
end while
```

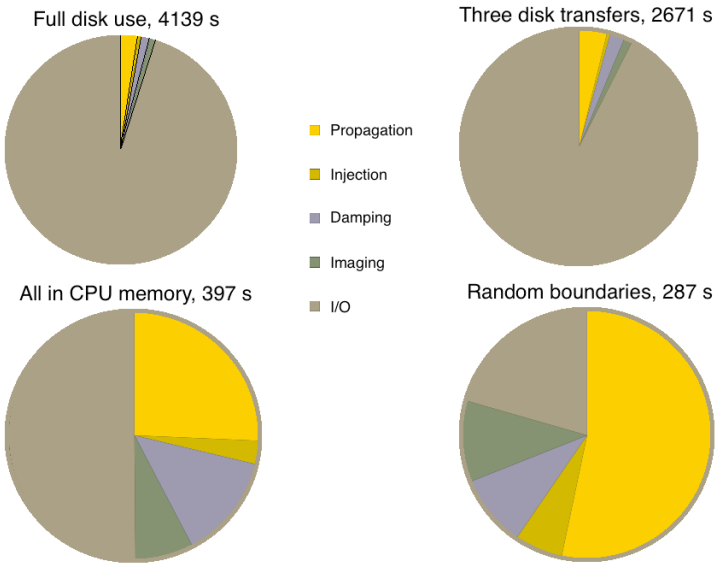

Figure 6.8: Visual representations of how I/O balances relative to other operations for four GPU based RTM scenarios: disk based wavefield access, asynchronous disk based wavefield access, CPU DRAM based wavefield access and GPU based random-boundary wavefield re-computation. [**NR**] chap6/. ratios

A valid measure of how well a system is being used is to look at how much time is spent on computation and I/O as a function of the total time. Figure 6.8 shows such a breakdown for four separate methods (all GPU based). These are disk based RTM, asynchronous disk based RTM, CPU DRAM based RTM and finally random boundary based RTM. Only the last of these is computationally dominated. Given that the random boundary noise is rarely a problem, it can be concluded that this adaptation is the best suited for the given computers.

Extension to multiple GPUs

The bulk of this research was carried out in 2011 and 2012 when the Fermi M2090 cards and CUDA 4.0 were the state-of-the-art in GPU computing. For the following discussion the compute nodes used each contained 8 Fermi M2090 GPUs, each with 6 Gbytes of global memory (48 Gbytes combined), and 24 CPU cores, each with with

8 Gbytes global memory (192 Gbytes combined).

For the most simple RTM implementation seven fields of significant (3D) size must be allocated on the GPU - two source wavefields, two receiver wavefields, the velocity model, the image estimate, and the data. Since an individual Fermi M2090 card possesses 6 Gbytes of memory this limits the potential model size to just over $600pts^3$. Many contemporary imaging targets are far larger than this, so an approach that can accommodate larger models must be used.

The fact there are multiple GPUs per node can be taken advantage of. They have a combined global storage of 48 Gbytes, making the new potential image size $1900pts^3$. Through domain decomposition the full combined memory space can be utilized. Given a large 3D volume there are many ways to perform domain decomposition; for now decomposition into equally sized blocks is assumed. Due to the 'tree' layout of how the GPUs are linked it only makes sense to split the domain along a single axis, then the overlapping region for a given sub-domain lies on the GPUs 'left' and 'right' of the current GPU. Furthermore if this decomposition is performed along the slowest axis then this overlapping region (needed for communication) is contiguous in memory. Figure 6.9 shows this for the situation where the y-axis is the slow axis. The domain is split evenly along y and the overlapping regions for, say, GPU 1, lie on GPU 0 and GPU 2. These sub-domains overlap by the stencil width, so that the convolution can be applied correctly across these boundaries.

How these regions overlap is shown in Figure 6.10, the overlap is equal to the stencil width and the half stencil width region is often referred to as the 'halo'. To apply the convolution each GPU needs its halo region to be updated; it receives these values from its neighboring GPU(s).

An additional benefit of CUDA 4.0 is the ability to perform calculations and memory transfer operations simultaneously. Thus, by staggering some of the calculation it is possible to almost entirely hide the communication (halo update) part of the propagation. Initially, each GPU will run a kernel that only calculates the values of the region that it's neighbors will need (their halo region.) So GPU 0 will calculate

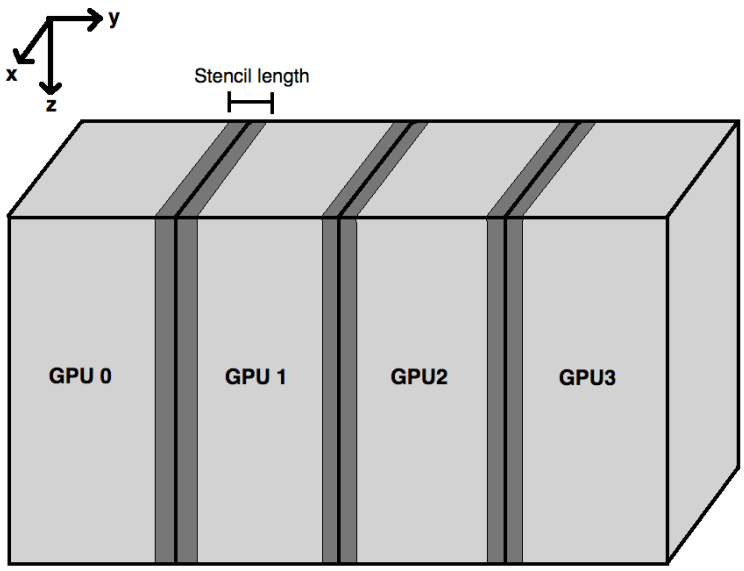

Figure 6.9: Breaking the computational domain across multiple GPUs, dark grey indicates the overlapped area between neighboring GPUs. [**NR**] chap6/. domdec

the halo region of GPU 1, GPU 1 will calculate the right-side halo region of GPU 0 and the left-side halo region of GPU 2 etc. (Figure 6.10). Once all these transfer regions have been calculated then a new set of kernels can start running to calculate the rest of the wavefield (internal regions). While the internal calculation is being performed a set of asynchronous memory copies can be initiated, which transfers the halo values between GPUs to their appropriate spots.

The connections between GPUs are known as PCIe links, and these are duplex. Thus they can send and receive at the same time, as long as the directions are opposite. GPU 1 can receive information from GPU 0 and send information to GPU 2 at the same time, but could not send and receive to GPU 0 at the same time. Consequently, to complete the full set of transfers initially each GPU will send to the right (higher GPU index) and receive from the left. Then, send left and receive right. After this has completed each GPU will have its halo regions updated to the correct values. If this communication time is less than the time to complete the internal calculation then the communication is effectively hidden - there is no waiting time involved with

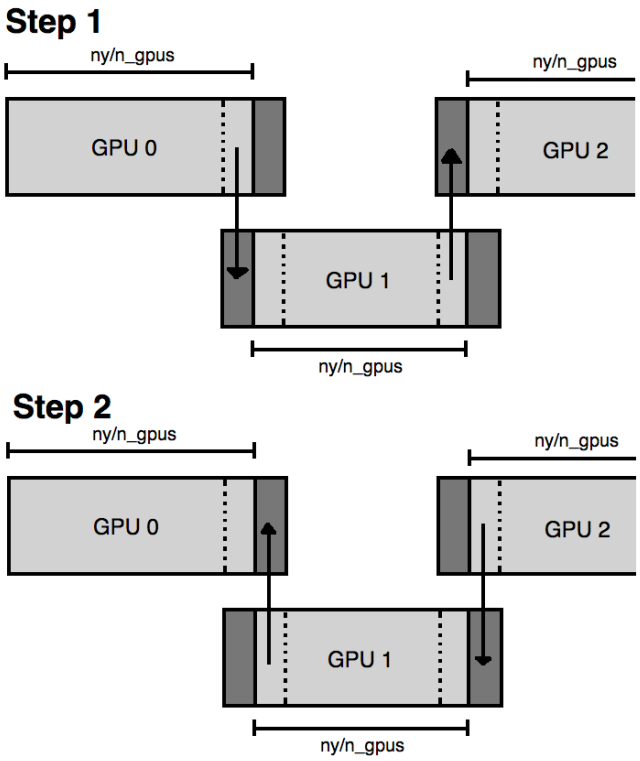

Figure 6.10: Staggered representation for halo region transfer. Initially all GPUs send to the 'right' and receive from the 'left,' then the order is reversed. [**NR**] chap6/. halos

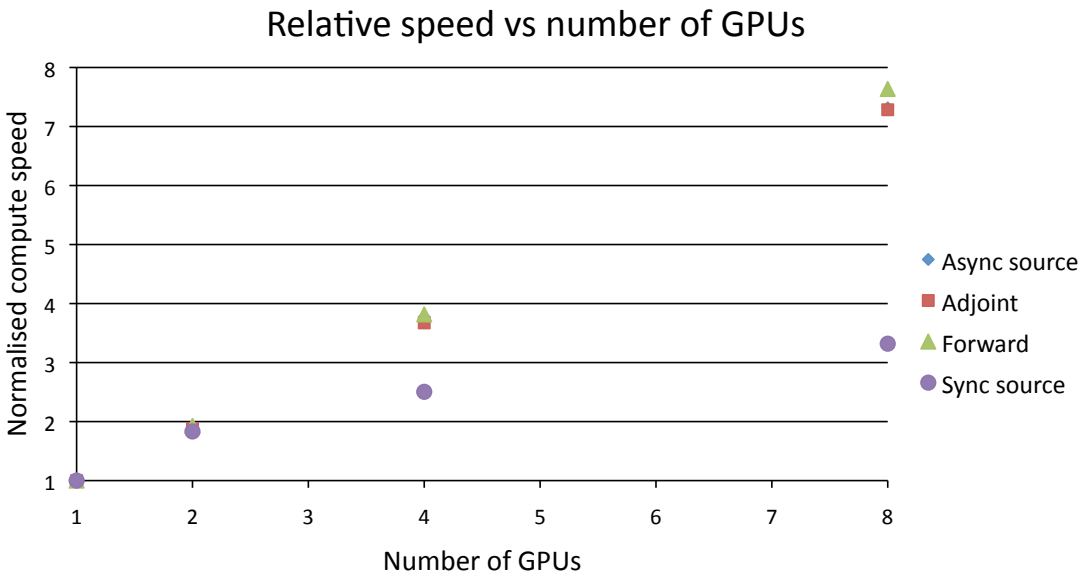

Figure 6.11: How the individual linearized inversion operations scale with the number of GPUs used, relative to normalized computational speeds. [**NR**] chap6/. speeds

the decomposition scheme. There is a small overhead cost associated with splitting the halo and internal calculations, but this is just a few percent. The sequence in abridged CUDA can be seen below for reference.

```
for(i_gpu=0; i_gpu < n_gpus; i_gpu++){
   cudaSetDevice(i_gpu);
   kernel<<<...halo_region[],halo_stream[i_gpu]>>>(...);
   kernel<<<...internal_region[],internal_stream[i_gpu]>>>(...);
}
for(i_gpu=0; i_gpu < n_gpus; i_gpu++){
   cudaMemcpyPeerAsync(...halo_stream[i_gpu]);
}
for(i_gpu=0; i_gpu < n\_gpus; i_gpu++){
   cudaStreamSynchronise(...halo_stream[igpu]);
}
```

```
for(i_gpu=0; i_gpu < n_gpus; i_gpu++){
   cudaMemcpyPeerAsync(...halo_stream[i_gpu]);
}
for(i_gpu=0; i_gpu < n_gpus; i_gpu++){
   cudaDeviceSynchronise();
}
```

To judge how well the communication-hidden domain decomposition works one can begin with a model size that fits on a single GPU. The same imaging procedure can then be used for two, four and eight GPUs and their normalized compute speeds can be plotted as function of the number of GPUs used. This is known as a 'stong scaling' test. Figure 6.11 shows the results. In purple the speed-ups observed without communication hiding are plotted. The speed up quickly saturates, similar to the naive OpenMP efforts discussed earlier. However, for both the propagation (blue) the full forward process (green) and the full adjoint process (red) the speed up is very close to linear. This means the communication has been effectively hidden and decomposing a problem across 8 GPUs will give close to 8x the computational speed.

BEYOND ACOUSTIC GPU PROPAGATION

The majority of this thesis study used acoustic wave propagation, to describe how energy moves through the Earth. As discussed in Chapter 5, the field dataset used was acquired over a section of the Earth where these acoustic assumptions were too inaccurate. The provided models described the target area as Tilted Transverse Isotropic (TTI), which results in a more computationally intensive propagation kernel, and many more significant allocations necessary.

The full mathematical description of TTI based propagation involved many details unnecessary to this thesis discussion, but an approach as described by Fletcher et al. (2009) was used, and a kernel written by Paulius Micikevicius was implemented.

The major issue with adapting this provided kernel was that of memory allocation.

The test example was over a very small model, where many 3D field allocations was not a concern. For acoustic propagation, seven large fields must be allocation - two receiver wavefield snapshots, two source wavefield snapshots, the image, the velocity model, and the recorded data. TTI propagation decomposes both sets of wavefields into two components, requires an intermediate derivative calucation for both wavefields, and uses four additional Earth model components. This results in eight allocations per wavefield, and a total of 23 significant allocations.

This many allocations over 8 GPUs, in terms of memory requirements, is analogous to four times the memory needed for acoustic propagation. Fortunately, a few tricks could be used to make 3D GPU based TTI propagation viable for such a large dataset.

Firstly, these aforementioned intermediate derivative fields are not updated between propagation steps, they depend only on the inputs for a given time. This means the computation for the source wavefield and the receiver wavefield can be performed sequentially, and the intermediate fields zeroed and re-used. This saves two allocations, and only increases propagation time by a fraction of a percent.

Secondly, the four anisotropic parameters do not require a large dynamic range for storage. The Thomsen parameters (Thomsen, 1986), ϵ and δ, vary between 0 and 0.3, with a required precision of two decimal places. The angular parameters, ϕ and θ, very between 0 and π, with a similar required precision. CUDA provides support for half-integer storage, and by storing these parameters in unsigned-short arrays, all four fields take up the equivalent space of two fields - saving another two allocations.

Thirdly, the full shot does not need to be allocated on the GPU at all times, only the necessary time slices for injection and propagation. This means the shot can be streamed as it is propagated, requiring much less GPU space. To account for the fact that the data is more coarsely sampled than the necessary propagation interval, an eight-point since interpolator is used. Thus, at the minimum, only eight time slices of the data need to be resident on the GPU. However, if only eight slices are used then there must be CPU-GPU transfers at every imaging time step, as the next data time-slice will need to be transferred. Instead, a block of, say, 100 slices can be

allocated on the GPU. Once these slices have all been propagated into the model, this array can be flushed and re-filled with the next hundred slices. This method increases overhead by a few percent, but makes the data allocation essentially negligible.

For the given dataset, these three techniques reduced the necessary allocations to the point where imaging was now viable.

EXTENDED BORN MODELING

Chapter 4 referenced that extended Born modeling posed an additional computation issue, when compared to extended RTM. This is, again, due to the asymmetry of the CPU-GPU compute nodes.

During extended imaging, there is some flexibility when it comes to implementation. Each shot is independent, as are the source and receiver wavefields, and the output image is stacked over time, t. This means, given a CPU-GPU node, it is possible to overlap the propagation of one shot with the imaging of another. The first shot, S_i, is propagated on the GPU, and the wavefields are saved (either to CPU DRAM or disk, depending on size.) The next shot, S_{i+1}, can begin propagation on the GPU. Concurrently, a series of CPU threads can perform the extended imaging for shot S_i. This overlapping of imaging and propagation can result in significant savings.

For extended Born modeling, the source and receiver wavefields are not independent. Propagating the receiver wavefield is contingent on the source wavefield, after convolution with the image. As a result it is not possible to overlap the convolution of one source with the propagation of another.

CONCLUSIONS

There are many options for accelerating linearized optimization, and the best approach depends strongly on the computational system at one's disposal. For a set of

multi-core CPU nodes then a straight OpenMP solution to accelerating the computation quickly saturates as a function of cores. By breaking the domain into sets of blocks to attempt a better cache performance a performance increase is seen, however it is far from the theoretical performance.

GPUs are a clear option since they are designed to perform with algorithms capable of fine-grain parallelization, such as time-domain finite-difference wave propagation. For single card performance fast speeds are observed for the propagator alone, but for use with RTM the source wavefield I/O problem must be addressed. Disk access during propagation can be mitigated by using a random boundary condition and favoring computation over communication. Such a scheme is the only implementation that is dominated by compute rather than I/O, giving excellent performance compared to the parallel CPU scheme.

The memory size of a single GPU is restrictive. By attaching several GPUs to a single node it is possible to break the computational domain across multiple GPUs, allowing larger models to be used. Such a method requires communication between regions between time steps, which can result in performance saturation with additional GPUs. By breaking the computation into multiple steps and using asynchronous memory transfers it is possible to entirely hide the communication, resulting in a close to linear speed up with additional GPUs.

ACKNOWLEDGMENTS

Special thanks to Paulius Micikevicius, formerly of NVIDIA, for sharing example codes and partaking in many insightful multi-GPU discussions.

Bibliography

Abma, R., T. Manning, M. Tanis, J. Yu, and M. Foster, 2010, High quality separation of simultaneous sources by sparse inversion: Presented at the 72nd EAGE Conference & Exhibition.

Abma, R., Q. Zhang, A. Arogunmati, G. Beaudoin, et al., 2012, An overview of bps marine independent simultaneous source field trials: 82nd Ann. Int. Mtg, SEG.

Abma, R. L., 2012, Method for separating independent simultaneous sources. (US Patent 8,295,124).

Abma, R. L. and A. Ross, 2014, Seismic source coding, activation, and acquisition. (US Patent App. 14/540,506).

Akerberg, P., G. Hampson, J. Rickett, H. Martin, J. Cole, et al., 2008, Simultaneous source separation by sparse radon transform: Presented at the 2008 SEG Annual Meeting.

Alewine, R. W., 1974, Application of linear inversion theory toward the estimation of seismic source parameters: PhD thesis, California Institute of Technology.

Alford, R., K. Kelly, and D. M. Boore, 1974, Accuracy of finite-difference modeling of the acoustic wave equation: Geophysics, **39**, 834–842.

Alkhalifah, T., 2000, An acoustic wave equation for anisotropic media: Geophysics, **65**, 1239–1250.

Alvarez, G. F., 2007, Attenuation of multiples in image space.: PhD thesis, Stanford University.

Amdahl, G. M., 1967, Validity of the single processor approach to achieving large scale computing capabilities: Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, 483–485, ACM.

Aravkin, A., M. P. Friedlander, F. J. Herrmann, and T. Van Leeuwen, 2012a, Robust inversion, dimensionality reduction, and randomized sampling: Mathematical Programming, **134**, 101–125.

Aravkin, A. Y., M. P. Friedlander, and T. van Leeuwen, 2012b, Robust inversion via semistochastic dimensionality reduction: Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on, 5245–5248.

Ayeni, G., Y. Tang, B. Biondi, et al., 2009, Joint preconditioned least-squares inversion of simultaneous source time-lapse seismic data sets: Presented at the 2009 SEG Annual Meeting.

Ayeni, G, T. Y. and B. Biondi, 2009, Joint preconditioned least-squares inversion of simultaneous source time-lapse seismic data sets: SEG Expanded Abstracts, **28**, 3914–3918.

Baysal, E., D. Kosloff, and J. Sherwood, 1983, Reverse time migration: Geophysics, **45**, 1514–1524.

Beasley, C. J., 2008, A new look at marine simultaneous sources: The Leading Edge, **27**, 914–917.

Beasley, C. J., R. E. Chambers, and Z. Jiang, 1998a, A new look at simultaneous sources: SEG Technical Program Expanded Abstracts, **68th**, 133–135.

Beasley, C. J., R. E. Chambers, Z. Jiang, et al., 1998b, A new look at simultaneous sources: SEG Technical Program Expanded Abstracts, 133–135.

Berkhout, A. ., 2008, Changing the mindset in seismic data acquisition: The Leading Edge, **27**, 924–938.

Boonyasiriwat, C., G. T. Schuster, et al., 2010, 3d multisource full-waveform inversion using dynamic random phase encoding: SEG Technical Program Expanded Abstracts, 1044–1049.

Boore, D. M., 1972, Finite difference methods for seismic wave propagation in heterogeneous materials: Methods in computational physics, **11**, 1–37.

Bouska, J. et al., 2009, Distance separated simultaneous sweeping: Efficient 3d vibroseis acquisition in oman: 79th SEG meeting, Houston, Texas, USA, Expanded Abstracts, 1–5.

Chapman, B., G. Jost, R. Van der Pas, and D. J. Kuck, 2008, Using openmp :

portable shared memory parallel programming: MIT Press.

Chauris, H. and M. Benjemaa, 2010, Seismic wave-equation demigration/migration: Geophysics, **75**, S111–S119.

Claerbout, J., 2001, Image estimation by example: Society of Exploration Geophysicists.

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

Claerbout, J. F. and J. Black, 2008, Basic earth imaging: Citeseer.

Clapp, M. L., 2005, Imaging under salt: illumination compensation by regularized inversion: PhD thesis, Stanford University.

Clapp, R., 2008, Reverse time migration with random boundaries: SEP-138, **138**.

Dablain, M., 1986, The application of high-order differencing to the scalar wave equation: Geophysics, **51**, 54–66.

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

Dai, W., K. Jiao, D. Coles, and R. Coates, 2014, Least-squares reverse-time migration with statistical sampling: Presented at the 76th EAGE Conference and Exhibition 2014.

Dash, R., G. Spence, R. Hyndman, S. Grion, Y. Wang, and S. Ronen, 2009, Wide-area imaging from obs multiples: Geophysics, **74**, Q41–Q47.

Denning, P. J., 2005, The locality principle: Commun. ACM, **48**, 19–24.

Doulgeris, P., A. Mahdad, and G. Blacquière, 2010, Separation of blended data by iterative estimation and subtraction of interference noise.

Fehler, M. and K. Larner, 2008, Seg advanced modeling (seam): Phase i first year update: The Leading Edge, **27**, 1006–1007.

Fletcher, R. F., P. Fowler, P. Kitchenside, U. Albertin, et al., 2005, Suppressing artifacts in prestack reverse time migration: Presented at the 2005 SEG Annual Meeting.

Fletcher, R. P., X. Du, and P. J. Fowler, 2009, Reverse time migration in tilted transversely isotropic (tti) media: Geophysics, **74**, WCA179–WCA187.

Foltinek, D., D. Eaton, J. Mahovsky, P. Moghaddam, and R. McGarry, 2009, Industry

scale reverse time migration on GPU hardware: SEG Technical Program Expanded Abstracts, **28**, 2789–2793.

Fomel, S., 2001, Three-dimensional seismic data regularization: PhD thesis, Citeseer.

Fox, R. L., 1971, Optimization methods for engineering design: Addison-Wesley Pub. Co.

Friedlander, M. P. and M. Schmidt, 2012, Hybrid deterministic-stochastic methods for data fitting: SIAM Journal on Scientific Computing, **34**, A1380–A1405.

Fung, J., F. Tang, and S. Mann, 2002, Mediated reality using computer graphics hardware for computer vision.: ISWC, 83–89, IEEE Computer Society.

Godwin, J., P. Sava, et al., 2010, Blended source imaging by amplitude encoding: Presented at the 2010 SEG Annual Meeting.

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

Guitton, A., B. Kaelin, and B. Biondi, 2006, Least-squares attenuation of reverse-time-migration artifacts: Geophysics, **72**, S19–S23.

Herrmann, F. J., Y. A. Erlangga, and T. T. Lin, 2009, Compressive simultaneous full-waveform simulation: Geophysics, **74**, A35–A40.

Hou, J. and W. W. Symes, 2015, An approximate inverse to the extended born modeling operator: Geophysics, **80**, R331–R349.

Huo, S., Y. Luo, P. Kelamis, et al., 2009, Simultaneous sources separation via multi-directional vector-median filter: Presented at the 2009 SEG Annual Meeting.

Ikelle, L. T., 2001, Multi-shooting approach to seismic modeling and acquisition. (US Patent 6,327,537).

———, 2010, Coding and decoding: Seismic data: The concept of multishooting, volume **39**: Elsevier.

Jack, I., B. Taylor, D. Howe, T. Allen, M. Foster, et al., 2008, Independent simultaneous sweeping-a method to increase the productivity of land seismic crews: Presented at the 2008 SEG Annual Meeting.

Jones, I. F. and I. Davison, 2014, Seismic imaging in and around salt bodies: Interpretation, **2**, SL1–SL20.

Komatitsch, D., G. Erlebacher, D. Göddeke, and D. Michéa, 2010, High-order finite-element seismic wave propagation modeling with mpi on a large gpu cluster: Journal of Computational Physics, **229**, 7692–7714.

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.

Krohn, C. and R. Neelamani, 2008, Simultaneous sourcing without compromise: Presented at the 70th EAGE Conference & Exhibition.

Kuehl, H., 2002, Least-squares wave-equation migration/inversion.

Leader, C., A. Almomin, and R. Clapp, 2014, Phase-encoded inversion using randomised sampling: Presented at the 76th EAGE Conference and Exhibition 2014.

Liu, F., R. H. Stolt, D. W. Hanson, R. S. Day, et al., 2002, Plane wave source composition: an accurate phase encoding scheme for prestack migration: 72nd Ann. Mtg., Soc. Expl. Geophys.

Mahdad, A., P. Doulgeris, and G. Blacquiere, 2012, Iterative method for the separation of blended seismic data: discussion on the algorithmic aspects: Geophysical Prospecting, **60**, 782–801.

Mallón, D. A., G. L. Taboada, C. Teijeiro, J. Touriño, B. B. Fraguela, A. Gómez, R. Doallo, and J. C. Mouriño, 2009, Performance evaluation of mpi, upc and openmp on multicore architectures: Proceedings of the 16th European PVM/MPI Users' Group Meeting on Recent Advances in Parallel Virtual Machine and Message Passing Interface, 174–184, Springer-Verlag.

Manavski, S. and G. Valle, 2008, Cuda compatible gpu cards as efficient hardware accelerators for smith-waterman sequence alignment.: BMC Bioinformatics, **9**.

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

Moczo, P., J. Kristek, M. Galis, P. Pazak, and M. Balazovjech, 2007, The finite-difference and finite-element modeling of seismic wave propagation and earthquake motion: Acta physica slovaca, **57**.

Moore, I., B. Dragoset, T. Ommundsen, D. Wilson, D. Eke, C. Ward, et al., 2008, Simultaneous source separation using dithered sources: Presented at the 2008 SEG

Annual Meeting.

Morton, S. A. and C. C. Ober, 1998, Faster shot-record depth migrations using phase encoding: Technical report, Sandia National Labs., Albuquerque, NM (United States).

Mulder, W., 2001, Higher-order mass-lumped finite elements for the wave equation: Journal of Computational Acoustics, **9**, 671–680.

Mulder, W. A. and R.-E. Plessix, 2004, A comparison between one-way and two-way wave-equation migration: Geophysics, **69**, 1491–1504.

Neelamani, R., C. E. Krohn, J. R. Krebs, J. K. Romberg, M. Deffenbaugh, and J. E. Anderson, 2010, Efficient seismic forward modeling using simultaneous random sources and sparsity: Geophysics, **75**, WB15–WB27.

Nemeth, T., 1996, Imaging and filtering by least-squares migration: PhD thesis, The University of Utah.

Nemeth, T., C. Wu, and G. T. Schuster, 1999, Least-squares migration of incomplete reflection data: Geophysics, **64**, 208–221.

Ober, C. C., L. A. Romero, and D. C. Ghiglia, 2000, Method of migrating seismic records: Technical report, SANDIA CORP.

Ohmer, J., F. Maire, and R. Brown, 2005, Implementation of kernel methods on the GPU: DICTA'05 Expanded Abstracts, **78**.

Pan, G., R. A. Phinney, and R. I. Odom, 1988, Full-waveform inversion of plane-wave seismograms in stratified acoustic media: Theory and feasibility: Geophysics, **53**, 21–31.

Pecholcs, P. I., S. K. Lafon, H. Al-Shammery, P. G. Kelamis, O. Winter, J.-B. Kerboul, and T. Klein, 2010, Over 40,000 vps per day–with real-time quality control: Opportunities and challenges: 80th Annual International Meeting, SEG, Expanded Abstracts, 111–114.

Pratt, R. G., 1999, Seismic waveform inversion in the frequency domain, part 1: Theory and verification in a physical scale model: Geophysics, **64**, 888–901.

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.

———, 2000, Phase encoding of shot records in prestack migration: Geophysics, **65**,

426–436.

Sava, P., B. Biondi, et al., 2003, Wave-equation migration velocity analysis by inversion of differential image perturbations: 73rd Ann. Internat. Mtg Soc. of Expl. Geophys., submitted.

Sava, P. and A. Guitton, 2005, Multiple attenuation in the image space: Geophysics, **70**, V10–V20.

Sava, P. and I. Vasconcelos, 2011, Extended imaging conditions for wave-equation migration: Geophysical Prospecting, **59**, 35–55.

Shubin, G. R. and J. B. Bell, 1987, A modified equation approach to constructing fourth order methods for acoustic wave propagation: SIAM Journal on Scientific and Statistical Computing, **8**, 135–151.

Sirgue, L., J. T. Etgen, U. Albertin, and S. Brandsberg-Dahl, 2010, System and method for 3d frequency domain waveform inversion based on 3d time-domain forward modeling. (US Patent 7,725,266).

Spitz, S., A. Pica, G. Hampson, et al., 2008, Simultaneous source separation: a prediction-subtraction approach: Presented at the 2008 SEG Annual Meeting.

Stefani, J., G. Hampson, and E. Herkenhoff, 2007, Acquisition using simultaneous sources: Presented at the 69th EAGE Conference & Exhibition.

Steihaug, T., 1983, The conjugate gradient method and trust regions in large scale optimization: SIAM Journal on Numerical Analysis, **20**, 626–637.

Symes, W., 2007, Reverse time migration with optimal check pointing: Geophysics, **72**, 213–221.

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

Tang, Y. and B. Biondi, 2009, Least square migration/inversion of blended data: SEG Expanded Abstracts, **28**, 2859–2863.

———, 2011, Target-oriented wavefield tomography using synthesized born data: Geophysics, **76**, WB191–WB207.

Tang, Y., B. Biondi, et al., 2009, Least-squares migration/inversion of blended data: SEG Technical Program Expanded Abstracts, 2859–2863.

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

———, 1984b, Linearized inversion of seismic reflection data: Geophysical prospecting, **32**, 998–1015.

Thomsen, L., 1986, Weak elastic anisotropy: Geophysics, **51**, 1954–1966.

Tsvankin, I. and L. Thomsen, 1995, Inversion of reflection traveltimes for transverse isotropy: Geophysics, **60**, 1095–1107.

van Leeuwen, T., A. Y. Aravkin, and F. J. Herrmann, 2011, Seismic waveform inversion by stochastic optimization: International Journal of Geophysics, **2011**.

Verschuur, D., A. Berkhout, et al., 2009, Target-oriented least-squares imaging of blended data: Presented at the 2009 SEG Annual Meeting.

Versteeg, R., 1994, The marmousi experience: Velocity model determination on a synthetic complex data set: The Leading Edge, **13**, 927–936.

Verwest, B. and D. Lin, 2007, Modelling the impact of wide-azimuth acquisition on subsalt imaging: Geophysics, **72**, 241–250.

Vigh, D., E. W. Starr, and J. Kapoor, 2009, Developing earth models with full waveform inversion: The Leading Edge, **28**, 432–435.

Weibull, W. W. and B. Arntsen, 2014, Reverse-time demigration using the extended-imaging condition: Geophysics, **79**, WA97–WA105.

Wong, M., 2014, Imaging with multiples by least-squares reverse time migraiton.: PhD thesis, Stanford University.

Woodward, M. J., 1992, Wave-equation tomography: Geophysics, **57**, 15–26.

Yoon, K., C. Shin, S. Suh, L. R. Lines, and S. Hong, 2003, 3d reverse-time migration using the acoustic wave equation: An experience with the seg/eage data set: The Leading Edge, **22**, 38–41.

Zhang, L., J. W. Rector III, G. M. Hoversten, et al., 2003, An acoustic wave equation for modeling in tilted ti media: 73rd Annual International Meeting, Society of Exploration Geophysics, Expanded Abstracts, 153–156.

Zhang, Y. and L. Duan, 2013, Seismic data processing including predicting multiples using a reverse time demigration. (US Patent App. 13/860,567).

Zhang, Y., J. Sun, S. Gray, et al., 2007, Reverse-time migration: amplitude and

implementation issues: Presented at the 2007 SEG Annual Meeting.