Acoustic and elastic least-squares two-way wave equation migration with exact adjoint operator

by

Linan Xu

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Geophysics

Department of Physics
University of Alberta

Abstract

A major problem in exploration seismology entails estimating subsurface structures and properties via linearized inversion. The problem is often called "least-squares migration" where seismic imaging is posed as an iterative least-squares problem. The iterative solution employs the method of conjugate gradients which requires two operators: the forward operator and the adjoint operator. This thesis investigates the design of the forward operator and its associated exact adjoint for both acoustic and elastic least-squares migration. The forward operator is derived using the Born approximation and Green's functions of the two-way wave equation.

A few key steps were followed to obtain an adjoint operator that has the exact adjoint formulation of the forward operator. I first derive the Born approximation and discretize Green's functions using the finite difference method, where I have adopted a staggered grid algorithm with stepping in the time domain. Then, a simple workflow was used to describe the discrete forward operator in terms of the concatenated multiplication of matrices. Finally, the exact adjoint operator is obtained by taking the transpose of the discrete forward operator. The adjointness of the forward and adjoint operators can be verified by the dot product test. Unlike the conventional adjoint operator derived via the discretization of continuous kernels, I observe that the proposed exact adjoint operator can pass the dot product test in machine precision, which implies that the forward and the proposed adjoint operator achieve sufficient accuracy to use conjugate gradients to solve the aforementioned problem of least-squares migration.

While the forward and exact adjoint operators are derived in the form of matrices, creating explicit matrices in numerical solvers does not provide a memory-efficient implementation. To deal with this memory issue, a matrix-free programming process is applied to develop the forward operator and its exact adjoint operator. Preconditioning operators are also investigated to solve the least-squares migration for extended shot-index images efficiently.

Finally, the proposed method is tested via synthetic examples. Compared with conventional migration techniques, least-squares migration (both acoustic and elastic) is capable of attenuating low-wavenumber artifacts, compensating for insufficient illumination, and increasing the resolution of seismic images. Elastic least-squares migration provides an additional benefit of suppressing multi-parameter cross-talk.

To my parents and professors

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor Dr. Mauricio D. Sacchi. When I was a fourth-year student, I was interested in full waveform inversion, a state-of-art technique. To help me obtain my research goal, Dr. Sacchi offered me a great long-term roadmap, which started with reverse time migration, an undergraduate project. At that point, the project was harder than I expected. Dr. Sacchi encouraged me to overcome the difficulties. After I had accomplished the project, Dr. Sacchi accepted me as his master student and continued to guide me in exploring least-squares two-way wave equation migration, an advanced migration technique. Without his encouragement, guidance and mentoring, I would not be able to complete my master studies.

Next, I highly appreciate teaching and research assistantship from the University of Alberta and Signal Analysis and Imaging Group (SAIG). I also would like to thank Canada Foundation for Innovations (CFI) and Major Science Initiatives (MSI) programs and the Center of Information Service and Technology (IST) at the University of Alberta for providing high-performance computing resources in this research. A special thank you is given to Kyle Dubitz and Masao Fujinaga who provided me prompt technical supports through my master program. Without these financial and technical supports, my research project would not have been possible.

I would also like to thank all of my colleagues at SAIG. I would like to thank Aaron Stanton, Wenlei Gao, Jinkun Cheng, Chen Ke, Nasser Kazami, Gian Matharu and Tianqi Hu for sharing with me their insights in our discussions. A special thank you is given to Jiarao Huang, who is a Ph.D. candidate in Electrical Computing Engineering. She greatly helped me with the editing of my thesis.

Finally, I am grateful to my mother, Qizhang Cao and my father, Xiaojie Xu. They worked very hard in China to pay for my expensive tuition fee and living cost in Canada. Thanks for their love. Especially, I want to express my deepest thanks to my father. As a senior research fellow of China's energy security studies, he has taught me how to become an excellent researcher. With him as my role model, I am determined to make more achievements in my research. Moreover, I would like to extend my gratitude to my relatives, Xiaoge Xu, Qilei Cao, and their families. Thanks for their love and care.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Introduction to least squares migration and motivation

Reflection seismology is a series of methods that use seismic reflections to reveal subsurface structures and properties. In the methods, man-made sources (e.g. dynamite, air gun or large vibratory systems) are required to excite the subsurface and sensors are used to measure the response reflected from the interior of the earth. Seismic migration is a method to relocate the response measured on the surface into the subsurface, thereby producing an image of the earth interior.

Migration operators can be derived under ray- and wave-based theoretical frameworks. For instance, the ray-based migration can be derived using Kirchhoff's integral formulation (Gray, 1986; Schneider et al., 1998; Sun et al., 2000) or linearized Born scattering with high-frequency ray-based approximations (Červený et al., 1982; da Costa et al., 1989; Lazaratos and Harris, 1990; Hill, 1990). Wave-based migration operators can be derived using one-way or two-way wave equation propagators. One-way wave equation migration is derived via the factorization of the acoustic wave equation (Gazdag, 1978; Gazdag and Sguazzero, 1984; Stoffa et al., 1990; Kessinger, 1992; Popovici, 1996). Today the most advanced technique for seismic migration adopts propagators based on two-way wave equations, which is frequently called reverse time migration or two-way wave equation migration(Baysal et al., 1983; Loewenthal and Mufti, 1983; McMechan, 1983; Whitmore, 1983; Levin, 1984). A two-way propagator entails solving the complete wave equation problem via the finite difference method, which is able to handle large lateral velocity variations. This advantage permits reverse time migration to overcome the dip angle limit of the typical migration methods based on one-way propagators (Hale, 1992).

In general, migration methods apply linear operators on the pre-processed seismic data to find the location where seismic reflection occurs (Claerbout, 1971). On the other hand, least-squares migration (LSM) has been proposed as a linear inverse problem, whose goal is to use constraint inversion to minimize migration artifacts. Least-squares migration solves the inverse problem for media perturbation that can be used to fit the seismic data (Keys and Weglein, 1983; Nemeth et al., 1999; Kühl and Sacchi, 2003; Clapp, 2005; Valenciano et al., 2006; Symes, 2008; Ji, 2009; Yao, 2013; Duan et al., 2016; Feng and Schuster, 2016; Yang et al., 2016). As a linear inverse problem, the least-squares migration first forms a linear relationship between pre-processed seismic data and the image via a forward modeling (de-migration) operator and then, the image that honors the observations is retrieved via a linear inversion algorithm (Lailly, 1983). Usually, least-squares migration is solved via iterative gradient-based optimization methods (e.g. steepest descent, conjugate gradients), which also permit us to iteratively mitigate the influence of dead traces and impose good feature (Ronen et al., 1995) in the final image via incorporating data space weights (Nemeth et al., 1999) and model space constraints (Kühl and Sacchi, 2003), respectively.

In this research, least squares two-way wave equation migration is solved by an efficient linear inversion algorithm: conjugate gradients (Hestenes and Stiefel, 1952). At the core of conjugate gradients, it is necessary to define the forward operator (denoted as $\mathbf{L}$) and to design its subsequent adjoint operator (denoted as $\mathbf{L}^T$). Moreover, the forward operator and its adjoint operator must be constructed in a way that the two operators behave like an exact adjoint pair because the exact adjoint property is an essential requirement for the method of conjugate gradients to guarantee that the Hessian matrix ($\mathbf{L}^T\mathbf{L}$) is symmetric. The design of the exact adjoint pair is extremely simple when the forward operator is given in terms of a matrix since the exact adjoint operator is merely the transpose of the forward operator in the time domain. However, the forward operators are never given in terms of a matrix but a computer program composed of thousands of lines of code, which makes it a non-trivial task to turn the program into its adjoint operator. In order to derive the exact adjoint formulation, the adjoint operator is designed carefully by first understanding the structure of the forward operator in terms of matrices. Apparently, these matrices are only used to guide the development of our codes. At the time of writing computer codes, the forward and adjoint paper are programmed via a matrix-free process.

The forward and adjoint operator in least-squares migration originates from acoustic (Tarantola, 1984) and elastic (Mora, 1988, 1987) seismic tomography. The forward operator is derived based on the generalized Born approximation (Lailly, 1983; Keys and Weglein, 1983), which consists of a response kernel and a wave simulation. The adjoint operator resembles imaging principle (Claerbout, 1971; Tarantola, 1984), which composed of an imaging condition and a reverse-time simulation. In the derivation of Tarantola (1984), the imaging condition is analogized as the adjoint of the response kernel, and the reverse-time simulation is considered as the adjoint of the normal wave simulation. Note that, in this case, Tarantola derived both the forward operator and its adjoint operator in the continuous domain. In the discrete domain, however, Ji (2009) observes that the reverse time simulation cannot pass dot-product test (Claerbout, 1992) with the normal wave simulation. The proper design of the forward-adjoint pair for least-squares reverse time migration has been thoroughly explained by Ji (2009) in the context of post-stack operators. Ji (2009) adopted a time-domain discrete approach and the language of linear algebra to derive the forward-adjoint pair. In a similar vein, Yao and Jakubowicz (2015) investigated a matrix formulation of least-squares reverse time migration in the frequency domain. Zhang et al. (2015) proposed a similar approach with Tarantola (1984) to derive a continuous forward and adjoint formulations and then, the two formulations are discretized to obtain forward and adjoint operators for least-squares reverse time migration. My approach differs from the authors as mentioned above in several ways. First, I utilize a time-domain discrete approach that follows the work of Ji (2009). I expand the latter to the prestack migrations (of both acoustic and elastic). That means this research will investigate the exact adjoint formulation of both response kernel and wave simulation in the aforementioned problem of seismic tomography. Besides, the study provides a strategy for applying preconditioning to extended shot-index images. I also point out that the *optimize then discretize* approach proposed by Zhang et al. (2015) could lead to operators that do not pass the dot product test (Claerbout, 1992) with sufficient accuracy for applying the conjugate gradients method. On the other hand, our design leads to the forward and adjoint operators that pass the dot product test in machine precision. The latter verifies a forward-adjoint operator ($\mathbf{L}^T\mathbf{L}$) that is symmetric and therefore, also invertible via the method of conjugate gradients. Finally, the proposed operators are adopted to show the effectiveness of both acoustic and elastic LSRTM.

1.2 Thesis outline

In **Chapter 2**, I design the forward operator and its adjoint operator for acoustic least-squares two-way wave equation migration. Derivation begins with the forward operator. First, the acoustic wave equations are linearized and incorporated with boundary conditions of the perfectly matched layers (PML). Secondly, I use finite difference method to discretize the forward operator and propose a simple workflow to describe the discrete operator in the form of matrices. Finally, taking the transpose of the forward operator, the exact adjoint operator is obtained. The exact adjointness of the proposed forward and its adjoint operator is verified by the dot-product test. Three examples are provided to illustrate the effectiveness of the proposed operators. In the first example, I use a single shot example to compare the proposed exact adjoint operator with reverse time migration in the aspect of image quality and data fitting performance. In the second and the third example, multiple-shot examples are provided to demonstrate the advantages of the least-squares two-way wave equation migration compared with reverse time migration regarding image illumination and resolution.

In **Chapter 3**, I follow the same strategy introduced in **Chapter 2** to find the forward and its exact adjoint operator for elastic least-squares two-way wave equation migration. The forward and its exact adjoint operators are adopted to solve the elastic least-squares migration. I provide two synthetic examples to show the benefit of the elastic least-squares migration compared with the elastic two-way wave equation migration. In the first example, a single-shot survey demonstrates the performance of the elastic least-squares migration in the aspects of data fitting and cross-talk suppression. In the second example, a multiple-shot example illustrates the benefit of the elastic least-squares migration regarding image illumination, resolution, and cross-talk artifact suppression.

In **Chapter 4**, I stress the importance of the exact adjoint operator in conjugate gradients. The proposed strategy to design the exact operator is summarized, and some important observations are mentioned. In the end, I discuss the limitations of the proposed approach and provide suggestions for future work.

CHAPTER 2

Acoustic least-squares two-way wave equation migration

2.1 Introduction

This chapter discusses the problem of least-squares migration for acoustic media. I first derive the forward operator using the Born approximation and two-way acoustic Green function. Then the adjoint operator is derived and finally, forward and adjoint operators are used in the method of conjugate gradients to obtain subsurface acoustic images.

2.2 Forward modelling via the acoustic Born approximation (de-migration operator)

Acoustic wave propagation in a media with Lame parameter $\lambda(x,z)$ and density $\rho(x,z)$ is described by the following set of partial differential equations

$$\begin{aligned}
\frac{\partial v_x}{\partial t} - \frac{1}{\rho}\frac{\partial p}{\partial x} &= 0 \\
\frac{\partial v_z}{\partial t} - \frac{1}{\rho}\frac{\partial p}{\partial z} &= 0 \\
\frac{\partial p}{\partial t} - \lambda(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}) &= \frac{\lambda}{\rho}\delta(x - x_s)\int_0^t src(\tau)d\tau
\end{aligned} \tag{2.1}$$

where $v_x(x,z,t)$, $v_z(x,z,t)$ are particle velocities in the x and z directions, respectively. The scalar field $p(x,z,t)$ denotes pressure. Last, the term $src(t)$ represents the seismic source at position $x = x_s$, $z = 0$. I assume that background values for the acoustic parameters are known and given by $\lambda_0(x,z)$ and $\rho_0(x,z)$. Similarly, I assume that responses to these background parameters can be numerically computed and are given by $v_{x0}(x,z,t), v_{z0}(x,z,t), p_0(x,z,t)$. Furthermore, I assume that unknown and known acoustic parameters are related via small additive perturbations such that $\lambda(x,z) = \lambda_0(x,z)+\delta\lambda(x,z)$ and $\rho_0(x,z) = \rho_0(x,z)+\delta\rho(x,z)$. Similarly I define wavefields in terms of responses to background parameters plus perturbations $v_x(x,z,t) = v_{x0}(x,z,t) + \delta v_x(x,z,t)$, $v_z(x,z,t) = v_{z0}(x,z,t) + \delta v_z(x,z,t)$ and $p(x,z,t) = p_0(x,z,t) + \delta p(x,z,t)$. I first substitute the expressions for λ and ρ and their associated fields in equation 2.1. Then after canceling terms involving multiplication of small perturbations in acoustic parameters and fields, I end up with the following two partial differential equations

$$\begin{aligned}
\frac{\partial v_{x0}}{\partial t} - \frac{1}{\rho_0}\frac{\partial p_0}{\partial x} &= 0 \\
\frac{\partial v_{z0}}{\partial t} - \frac{1}{\rho_0}\frac{\partial p_0}{\partial z} &= 0 \\
\frac{\partial p_0}{\partial t} - \lambda_0(\frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z}) &= \frac{\lambda_0}{\rho_0}\delta(x-x_s)\int_0^t src(\tau)d\tau
\end{aligned} \tag{2.2}$$

and

$$\begin{aligned}
\frac{\partial(\delta v_x)}{\partial t} - \frac{1}{\rho_0}\frac{\partial(\delta p)}{\partial x} &= (\frac{1}{\rho_0^2}\frac{\partial p_0}{\partial x})\delta\rho \\
\frac{\partial(\delta v_z)}{\partial t} - \frac{1}{\rho_0}\frac{\partial(\delta p)}{\partial z} &= (\frac{1}{\rho_0^2}\frac{\partial p_0}{\partial x})\delta\rho \\
\frac{\partial(\delta p)}{\partial t} - \lambda_0(\frac{\partial(\delta v_x)}{\partial x} + \frac{\partial(\delta v_z)}{\partial z}) &= (\frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z})\delta\lambda
\end{aligned} \tag{2.3}$$

Equation 2.2 describes propagation in the known background media. I solve this equation using finite differences for a given source function. In the migration jargon, I will say that this equation computes the *source-side wavefield*. Clearly, the solution of equation 2.2 are fields given in terms of snapshots $v_{x0}(x,z,t)$, $v_{z0}(x,z,t)$, $p_0(x,z,t)$ that correspond to an experiment with a source at $x = x_s$. These wave fields will be considered known and part of our forward (demigration) operator. Equation 2.3 is equivalent to equation 2.2 but now sources are given by perturbation terms (right hand side of equation 2.3) that contain unknown media perturbations $\delta\lambda$ and $\delta\mu$. The perturbation wavefield snapshots ($\delta v_{x0}(x,z,t)$, $\delta v_{z0}(x,z,t)$, $\delta p_0(x,z,t)$) are also computable via finite differences. In other words, the finite difference method will be applied to both 2.2 and 2.3 to compute background fields resulting from a source at $x = x_s$ and responses to media perturbations $\delta\lambda$ and $\delta\rho$, respectively. I stress that in both cases propagation occurs in the media with known background parameters $\lambda_0(x,z)$ and $\rho_0(x,z)$.

Following with our analysis, the right-hand side of equation 2.3 can be written in matrix form as follows

$$\begin{pmatrix} r_{v_x} \\ r_{v_z} \\ r_p \end{pmatrix} = \begin{pmatrix} \frac{1}{\rho_0^2}\frac{\partial p_0}{\partial x} & 0 \\ \frac{1}{\rho_0^2}\frac{\partial p_0}{\partial x} & 0 \\ 0 & \frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z} \end{pmatrix} \begin{pmatrix} \delta\rho \\ \\ \delta\lambda \end{pmatrix} \tag{2.4}$$

The fields $r_{v_x}(x,z,t)$, $r_{v_z}(x,z,t)$ and $r_p(x,z,t)$ can be interpreted as responses to scatterers where $\delta\rho \neq 0$ or $\delta\lambda \neq 0$. From the equation 2.4, compressional modulus λ and density ρ are treated as two independent variables. Density can be expressed as a function of compressional modulus based on Gardner's relation (Gardner et al., 1974). In other words, the compressional modulus and density could be be regularized via lithologic constrains. However, the objective of this research is not to estimate the relation between compressional modulus and density. Instead, the proposed least-squares migration focuses on structural imaging. I have preferred to limit the research to estimation of one high-quality single parameter image because the single parameter inversion avoids unwanted artifacts introduced by multi-parameter crosstalk. It is important to stress, however, that in Chapter 3 I will discuss multi-parameter inversion. Removing density from our problem leads to an algorithm that resembles classical imaging techniques that are adopted for structural migration. Therefore, if the proposed least-squares migration is limited to solve for perturbation of $\delta\lambda$ and eliminates $\delta\rho$ in equation 2.4. Equation 2.4 becomes

$$\underbrace{r_p}_{\mathbf{r}} = \underbrace{(\frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z})}_{K} \underbrace{\delta\lambda}_{m} \tag{2.5}$$

Equation 2.5 is valid for each subsurface point x, z and all time t. One can generalize the above equation for all the subsurface and times via a matrix formulation. To this end I define a model grid to describe λ and ρ. The grid contains $nx \times nz$ points. Equation 2.5 becomes

$$\mathbf{r} = \mathbf{KSm} \tag{2.6}$$

where $\mathbf{m} \in R^{nx\,nz \times 1}$ represents the discretized acoustic parameters: $\mathbf{m} = \delta\boldsymbol{\lambda}$. Similarly, $\mathbf{r} \in R^{nx\,nz\,nt \times 1}$ represents source term of pressure which is associated to the perturbation $\mathbf{m}$. The matrix $\mathbf{S}$ is a spraying operator that copies the vector $\mathbf{m}$ to all times, $\mathbf{S} \in R^{nt\,nx\,nz \times\, nx\,nz}$ and finally, $\mathbf{K} \in R^{nx\,nz\,nt \times\, nx\,nz\,nt}$ is a diagonal matrix with elements containing derivatives of the source-side solution (equation 2.5). The solution of equation 2.3 can also be written in matrix form. The excitation source is given by the vector $\mathbf{r}$ in equation 2.6. The field $\delta u(x,z,t) = [\delta v_x(x,z,t), \delta v_z(x,z,t), \delta p(x_z,t)]^T$ is written for all discrete times and all spatial positions as $\delta\mathbf{u} \in R^{3\,nt\,nx\,nz\times 1}$. The solution of equation 2.3 in terms of finite differences can now be described as follows

$$\delta\mathbf{u} = \mathbf{G}\mathbf{r}, \tag{2.7}$$

where $\mathbf{G}$ is the matrix that represents the finite difference code. Finally, seismic observations can be computed by extracting data from $\delta\mathbf{u}$ via a sampling operator $\mathbf{R} \in R^{N\times 1}$ where N indicates total number of data points. Putting it all together, I have observations (observed scattered field) in terms of acoustic model perturbations

$$\begin{aligned}\mathbf{d}^{obs} &= \mathbf{R}\delta\mathbf{u} \\ &= \mathbf{R}\mathbf{G}\mathbf{r} \\ &= \mathbf{R}\mathbf{G}\mathbf{K}\mathbf{S}\mathbf{m}\,. \end{aligned} \tag{2.8}$$

Notice that in our formulation the sampling operator $\mathbf{R}$ could measure particle velocities and/or pressure. I can use a compact format to represent the problem in terms of a linear operator $\mathbf{L} = \mathbf{RGKS}$

$$\mathbf{d}^{obs} = \mathbf{L}\mathbf{m}\,. \tag{2.9}$$

The analysis so far is valid for one source but it can be easily generalized to surveys with more than one source. Equation 2.9 defines the forward problem for least-squares migration.

2.3 Adjoint operator and regularized least-squares linearized inversion

I have defined a single-parameter linearized inversion problem where the unknown is $\mathbf{m} = \delta\boldsymbol{\lambda}$. Let us assume I have observed data (scattered wavefield) and I would like to determine an estimate of the model perturbation vector $\mathbf{m}$. The simplest solution entails using the adjoint or transpose operator. The latter also corresponds to applying migration to preprocessed field observations

$$\hat{\mathbf{m}} = \mathbf{L}^T\mathbf{d}^{obs} \tag{2.10}$$

one can replace equation 2.9 into 2.10 to obtain

$$\hat{\mathbf{m}} = \mathbf{L}^T\mathbf{L}\mathbf{m}\ . \tag{2.11}$$

The last equation clearly shows that the image obtained via migration is not equal to the unknown vector $\mathbf{m}$ unless the migration operator is the inverse of the forward modelling operators. Therefore, I can claim that the migrated image $\hat{\mathbf{m}}$ is a distorted version of the unknown vector of perturbations $\mathbf{m}$. It is also interesting to note that in the presence of noise and operator mismatch one should solve the problem $\mathbf{Lm} \approx \mathbf{d}^{obs}$. In this case I write the estimation of $\mathbf{m}$ as an optimization problem where I minimize a cost function of the form

$$J = \|\mathbf{W}_d(\mathbf{Lm} - \mathbf{d}^{obs})\|_2^2 + \mu^2 \|\mathbf{W}_m \mathbf{m}\|_2^2 \,. \tag{2.12}$$

The first term of the cost function is the misfit. I have also equipped the misfit with a data space matrix of weights to account for differences in data quality and for missing observations (Kühl and Sacchi, 2003). The second term is the quadratic regularization term that contains model space weights $\mathbf{W}_m$ to penalize roughness in the solution. The positive scalar μ^2 is the tradeoff parameter that is used to emphasize the importance of regularization versus misfit. The solution that minimizes J is found using the method of conjugate gradients. The latter is a semi-iterative technique that construct the solution that minimize J in a series of steps (iterations). Each iteration requires the application of products of the form $\mathbf{La}$ and $\mathbf{L}^T\mathbf{b}$ where $\mathbf{a}$ and $\mathbf{b}$ are vectors of size $nx\, nz \times 1$ and $N \times 1$, respectively. One needs to understand the precise structure of $\mathbf{G}$ in $\mathbf{L}$ in order to design the adjoint operator $\mathbf{L}^T$ that is required by the conjugate gradients method to iteratively estimate $\mathbf{m}$. The adjoint is given by

$$\mathbf{L}^T = \mathbf{S}^T \mathbf{K}^T \mathbf{G}^T \mathbf{R}^T \,. \tag{2.13}$$

The operator $\mathbf{S}^T$ entails summation over time (adjoint of time spraying). Similarly, $\mathbf{R}^T$ inserts zeros in all grid points with no observations (adjoint of wavefield sampling). Similarly, the operator $\mathbf{K}^T$ can be computed with no effort as it multiplication by a diagonal matrix which turns to be equivalent to element-to-element multiplication with the precomputed divergence of the *source-side velocity wavefield* (equation 2.6). In the next section, I provide an algorithm to compute $\mathbf{G}$ and its adjoint $\mathbf{G}^T$. It is basically the algorithm that permits to solve the systems given by equations 2.2 and 2.3 via finite differences but expressed in terms of matrices.

2.4 Matrix-based forward operator G

In this section, I present the solution of the partial differential equations given by equations 2.2 or 2.3 in terms of finite differences and I explain the algorithm in terms of matrices. I will add extra complexity to the problem by incorporating perfectly matched layer (PML) boundary conditions to absorb artificial boundary reflections (Berenger, 1996). In order to apply the PML method the pressure field is virtually decomposed into two components. Then, the problem given by equation 2.3 becomes

$$\begin{cases} \frac{\partial \delta v_x}{\partial t} + \kappa_x \delta v_x = \frac{1}{\rho}\frac{\delta \partial p}{\partial x} + r_{vx} \\ \frac{\partial \delta v_z}{\partial t} + \kappa_z \delta v_z = \frac{1}{\rho}\frac{\delta \partial p}{\partial z} + r_{vz} \\ \frac{\partial \delta p_x}{\partial t} + \kappa_x \delta p_x = \lambda \frac{\delta \partial v_x}{\partial x} + \frac{1}{2} r_{px} \\ \frac{\partial \delta p_z}{\partial t} + \kappa_z \delta p_z = \lambda \frac{\delta \partial v_z}{\partial z} + \frac{1}{2} r_{pz} \\ \delta p = \delta p_x + \delta p_z \end{cases} \tag{2.14}$$

where the source terms $r_{[.]}$ can be used to describe a point source as in equation 2.2 or scatterers as in the partial differential equations obtained for the field perturbations (equation 2.3). Partial differential equations are discretized via the finite difference method adopting a staggered grid algorithm (Dablain, 1986; Levander, 1988; Yao, 2013). The discretized finite difference update can be expressed in the following matrix form

$$\begin{cases} \delta \mathbf{v}_x^{n+1} &= \mathbf{A}_1 \delta \mathbf{v}_x^n + \mathbf{B}_1 \mathbf{D}_1 \delta \mathbf{p}^n + \mathbf{r}_{vx}^n \\ \delta \mathbf{v}_z^{n+1} &= \mathbf{A}_2 \delta \mathbf{v}_z^n + \mathbf{B}_2 \mathbf{D}_2 \delta \mathbf{p}^n + \mathbf{r}_{vz}^n \\ \delta \mathbf{p}_x^{n+1} &= \mathbf{A}_3 \delta \mathbf{p}_x^n + \mathbf{B}_3 \mathbf{D}_3 \delta \mathbf{v}_x^{n+1} + \frac{1}{2} \mathbf{r}_{px}^n \\ \delta \mathbf{p}_z^{n+1} &= \mathbf{A}_4 \delta \mathbf{p}_z^n + \mathbf{B}_4 \mathbf{D}_4 \delta \mathbf{v}_z^{n+1} + \frac{1}{2} \mathbf{r}_{pz}^n \\ \delta \mathbf{p}^{n+1} &= \delta \mathbf{p}_x^{n+1} + \delta \mathbf{p}_z^{n+1} \end{cases} \tag{2.15}$$

where $n = 1 \dots nt$ indicates time step. Notice, each vector in equation 2.15 is size $nx\,nz \times 1$ and corresponds to a spatial field at time n. The matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ are diagonal matrices (see Appendix A1). Matrices $\mathbf{D}_i$ are first order spatial derivative stencils operating on vectorized fields. These matrices can be easily obtained via Kronecker products. Furthermore, equation 2.15 can be written as follows

$$\delta \mathbf{u}^{n+1} = \mathbf{T} \delta \mathbf{u}^n + \mathbf{r}^{n+1} \tag{2.16}$$

where

$$\delta \mathbf{u}^{n+1} = \begin{bmatrix} \delta \mathbf{v}_x \\ \delta \mathbf{v}_z \\ \delta \mathbf{p}_x \\ \delta \mathbf{p}_z \\ \delta \mathbf{p} \end{bmatrix}^{n+1} \qquad \delta \mathbf{u}^{n} = \begin{bmatrix} \delta \mathbf{v}_x \\ \delta \mathbf{v}_z \\ \delta \mathbf{p}_x \\ \delta \mathbf{p}_z \\ \delta \mathbf{p} \end{bmatrix}^{n} \qquad \delta \mathbf{r}^{n+1} = \begin{bmatrix} \delta \mathbf{r}_{vx} \\ \delta \mathbf{r}_{vz} \\ \delta \mathbf{r}_{px} \\ \delta \mathbf{r}_{pz} \\ \mathbf{0} \end{bmatrix}^{n+1}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{A}_3 & \mathbf{0} & \mathbf{B}_3\mathbf{D}_3 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_4 & \mathbf{0} & \mathbf{B}_4\mathbf{D}_4 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_1\mathbf{D}_1 \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{0} & \mathbf{0} & \mathbf{B}_2\mathbf{D}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{2.17}$$

with $\delta\mathbf{u}^0 = 0$. For simplicity, I consider a simulation for $n = 1, 2, 3$ where one injects the source $\mathbf{r}$ to estimate snapshots for all wavefields in $\delta\mathbf{u}$

$$\delta\mathbf{u}^0 = \mathbf{0} \tag{2.18}$$

$$\delta\mathbf{u}^1 = \mathbf{r}_1 \tag{2.19}$$

$$\delta\mathbf{u}^2 = \mathbf{T}\,\mathbf{r}_1 + \mathbf{r}_2 \tag{2.20}$$

$$\delta\mathbf{u}^3 = \mathbf{T}^2\,\mathbf{r}_1 + \mathbf{T}\,\mathbf{r}_2 + \mathbf{r}_3 \tag{2.21}$$

$$\delta\mathbf{u}^4 = \mathbf{T}^3\,\mathbf{r}_1 + \mathbf{T}^2\,\mathbf{r}_2 + \mathbf{T}\,\mathbf{r}_3 + \mathbf{r}_4\,. \tag{2.22}$$

The last system of equation can be expressed as follows

$$\begin{bmatrix} \delta\mathbf{u}^1 \\ \delta\mathbf{u}^2 \\ \delta\mathbf{u}^3 \\ \delta\mathbf{u}^4 \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{T}^2 & \mathbf{T} & \mathbf{I} & \mathbf{0} \\ \mathbf{T}^3 & \mathbf{T}^2 & \mathbf{T} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{r}^1 \\ \mathbf{r}^2 \\ \mathbf{r}^3 \\ \mathbf{r}^4 \end{bmatrix} \tag{2.23}$$

which can also be written in terms of 3 steps

$$\begin{bmatrix} \delta\mathbf{u}^1 \\ \delta\mathbf{u}^2 \\ \delta\mathbf{u}^3 \\ \delta\mathbf{u}^4 \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}}_{\mathbf{G}} \begin{bmatrix} \mathbf{r}^1 \\ \mathbf{r}^2 \\ \mathbf{r}^3 \\ \mathbf{r}^4 \end{bmatrix} \tag{2.24}$$

I have now been able to construct our matrix $\mathbf{G}$ that propagates all the fields in $\mathbf{r}^n$ to responses $\delta\mathbf{u}^n$ for $n = 1 \dots nt$. Clearly, I have used $nt = 4$ to avoid writing pages and pages of matrices.

2.5 The exact adjoint operator

The adjoint of $\mathbf{G}$ is not as simple as one might have thought. Consider equation 3.25, I first need to flip the order of each of the sub-matrices to compute the adjoint and then, I need to transpose each sub-matrix. If one only flips the order of the matrices that form $\mathbf{G}$, I have the pseudo-adjoint operator of $\mathbf{G}$

$$\mathbf{G}^{pa} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{T} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{T} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{T} & \mathbf{I} \end{bmatrix}. \qquad (2.25)$$

The code that applies the pseudo-adjoint is equivalent to run the finite differences solver backwards ($n = nt \dots 1$). It reduces to inject the perturbation $\delta \mathbf{u}^n$ to obtain $\mathbf{r}^n$. On the other hand, the true adjoint also entails taking the transpose of all sub-matrices in $\mathbf{G}$

$$\mathbf{G}^T = \begin{bmatrix} \mathbf{I} & \mathbf{T}^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{T}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{T}^T \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}. \qquad (2.26)$$

I stress that $\mathbf{G}^{pa}$ and $\mathbf{G}^T$ are different as the results of having introduced boundary conditions (matrices $\mathbf{A}_i$ and $\mathbf{B}_i$ in 2.17). One could run a reverse time migration algorithm by either using the true adjoint

$$\mathbf{m}_{adj} = \mathbf{S}^T \mathbf{K}^T \mathbf{G}^T \mathbf{R}^T \, \mathbf{d}^{obs} \qquad (2.27)$$

or, the pseudo-adjoint operator

$$\mathbf{m}_{pa} = \mathbf{S}^T \mathbf{K}^T \mathbf{G}^{pa} \mathbf{R}^T \, \mathbf{d}^{obs} \, . \qquad (2.28)$$

In our experience, applying the true adjoint or the pseudo adjoint makes almost no difference for classical migration. However, when computing least-squares migration solutions via the method of conjugate gradients, one must adopt the exact adjoint propagator $\mathbf{G}^T$. In other words, $\mathbf{G}^{pa}\mathbf{G}$ is non-symmetric and consequently, the method of conjugate gradients will fail to converge (Hestenes and Stiefel, 1952).

Last, I point out that the matrix formulation adopted in this work helps us derive a forward-adjoint pair that rigorously passes the dot product test. It is unrealistic, however, to save large matrices in memory. In practice, I directly work with operators that were programmed in C programming language via a series of spatial and temporal loops. The matrix formulation of the forward-adjoint pair has allowed us to identify the parts of our code that required special attention in order to pass the dot product test.

2.6 Numerical examples

I present three numerical examples to explore least-squares migration with two-way wave equation operators. Synthetic data were generated by the finite difference method with PML boundary conditions.

2.6.1 Example 1: SAIG velocity model

I first compare our adjoint operator with the classical reverse time migration code that uses the pseudo-adjoint. The model for this particular numerical test consists of an anticlinal shown in Figure 2.1. The size of the velocity model is 300×750 grid points in the vertical and horizontal coordinates, respectively. The spatial sampling for both coordinates is 10m. I have generated a data set that corresponds to one common shot gather at position $x = 3.75\,\text{km}$. The velocity field ranges from 1.5km/s to 3.0km/s. Figures 2.2a and b portray images computed via the pseudo-adjoint and exact adjoint operators, respectively. The two results are extremely similar, in fact, the true adjoint and the pseudo-adjoint operators are fully interchangeable in the classical (non-iterative) reverse time migration.

I have also applied the method of conjugate gradients with preconditioning to estimate the seismic image. To this end, I minimize the following cost function

$$J = \|\mathbf{W}_d(\mathbf{LPu} - \mathbf{d}^{obs})\|_2^2 + \mu^2\|\mathbf{u}\|_2^2 \tag{2.29}$$

where

$$\mathbf{m} = \mathbf{Pu}\,. \tag{2.30}$$

In the data domain, preconditioning operator is denoted as $\mathbf{W}_d$, which is a diagonal matrix used to remove direct wave. In the model domain, preconditioning was introduced via a simple change of variable. The operator $\mathbf{P}$ contains diagonal weights that are proportional to the inverse of the source-side wavefield (Valenciano and Biondi, 2003; Guitton et al., 2006). I also mention that equations (29)-(30) correspond to reducing the cost function in equation (12) to its standard form by defining $\mathbf{P} = \mathbf{W}_m^{-1}$ (Ronen et al., 1995). In essence, I have transformed a least-squares problem with regularization term given by $\mathbf{W}_m$ into a similar problem with preconditioning operator $\mathbf{P}$. Often, the geophysical literature calls $\mathbf{W}_m$ the *bad-pass* operator. Whereas, $\mathbf{P}$ is called the *good-pass* operator. This emphasizes the high-pass and low-pass nature of $\mathbf{W}_m$ and $\mathbf{P}$, respectively. This will become evident in our next example where I use a low-pass preconditioning operator to minimize artifacts in extended images.

The cost of adding the preconditioning term $\mathbf{P}$ to our problem is minimal because the source-side wavefield was already precomputed and saved. In this particular case $\mathbf{P}$ compensates for illumination by normalizing the amplitudes of the image in terms of the source-side energy. Figure 2.3a shows the image computed via the method of conjugate gradients with the pseudo-adjoint operator. The solution corresponds to 14 iterations. It is important to mention that the solution starts to diverge after about 14 iterations. Preconditioning does not avoid the divergence of the conjugate gradients algorithm. In fact, divergence is expected because the forward and pseudo-adjoint operators do not constitute an exact adjoint pair (the migration-demigration operator is non-symmetric). For completeness, I have added converge curves given in terms of relative misfit reduction in Figure 2.4. This figure confirms that the method of conjugate gradients does not converge when the pseudo-adjoint operator is adopted. It also shows the benefit of adding preconditioning to the conjugate gradients method. Finally, Figure 2.5 confirms that the data is properly fitted after 14 iterations. The misfit at this point was reduced to about 1% of the misfit at the first iteration.

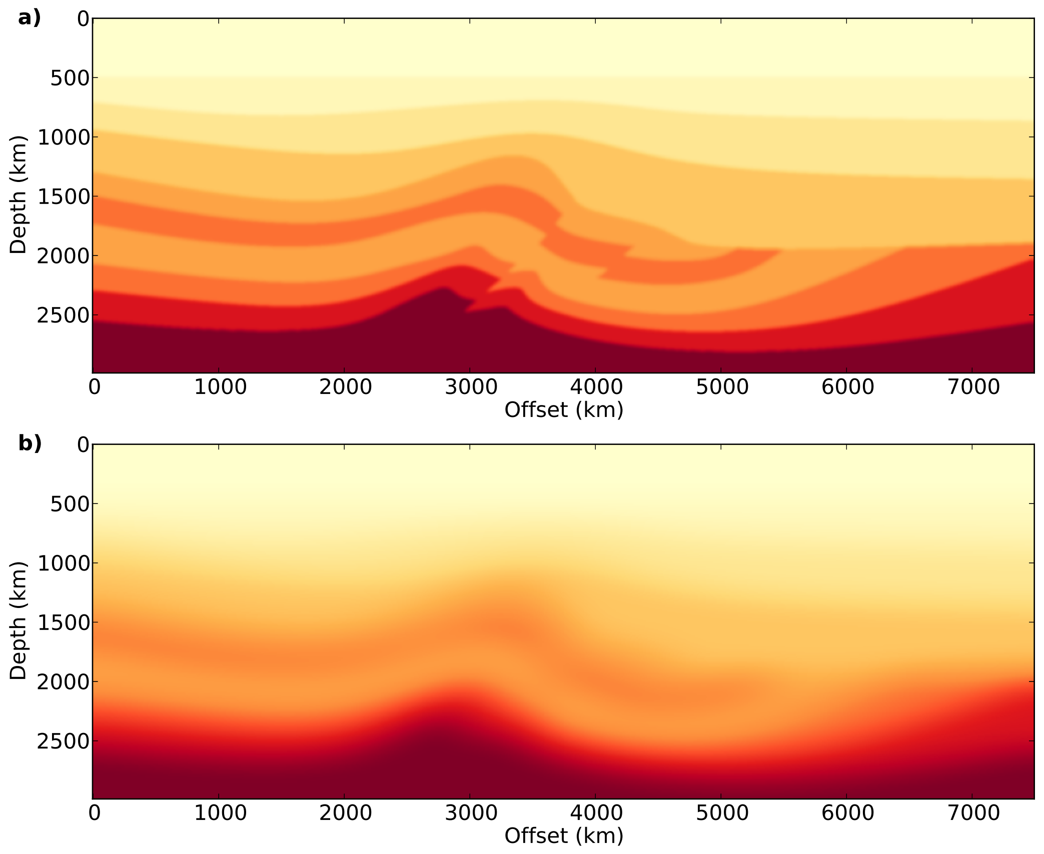

Figure 2.1: (a) True velocity model. (b) Smooth velocity model used to test least-squares migration.

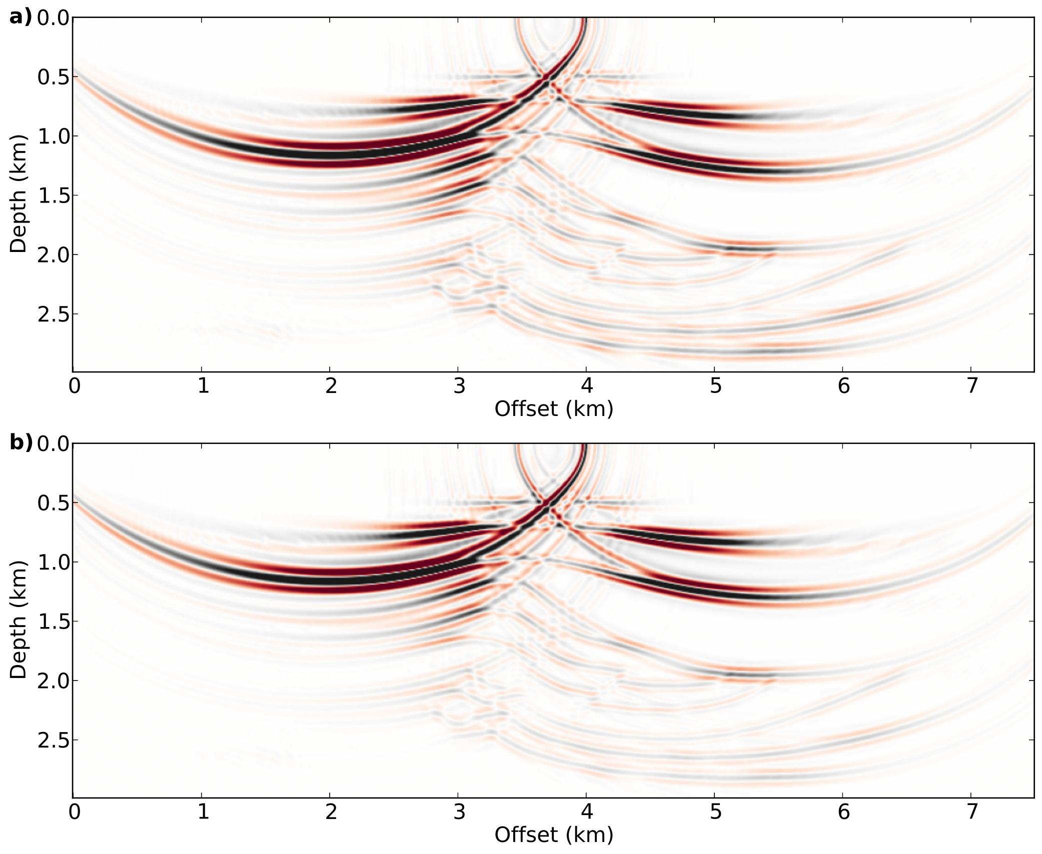

Figure 2.2: Our first example corresponds to the migration of one shot gather. (a) Migration via classical reverse time migraiton (pseudo-adjoint operator). (b) Migration via the exact adjoint operator.

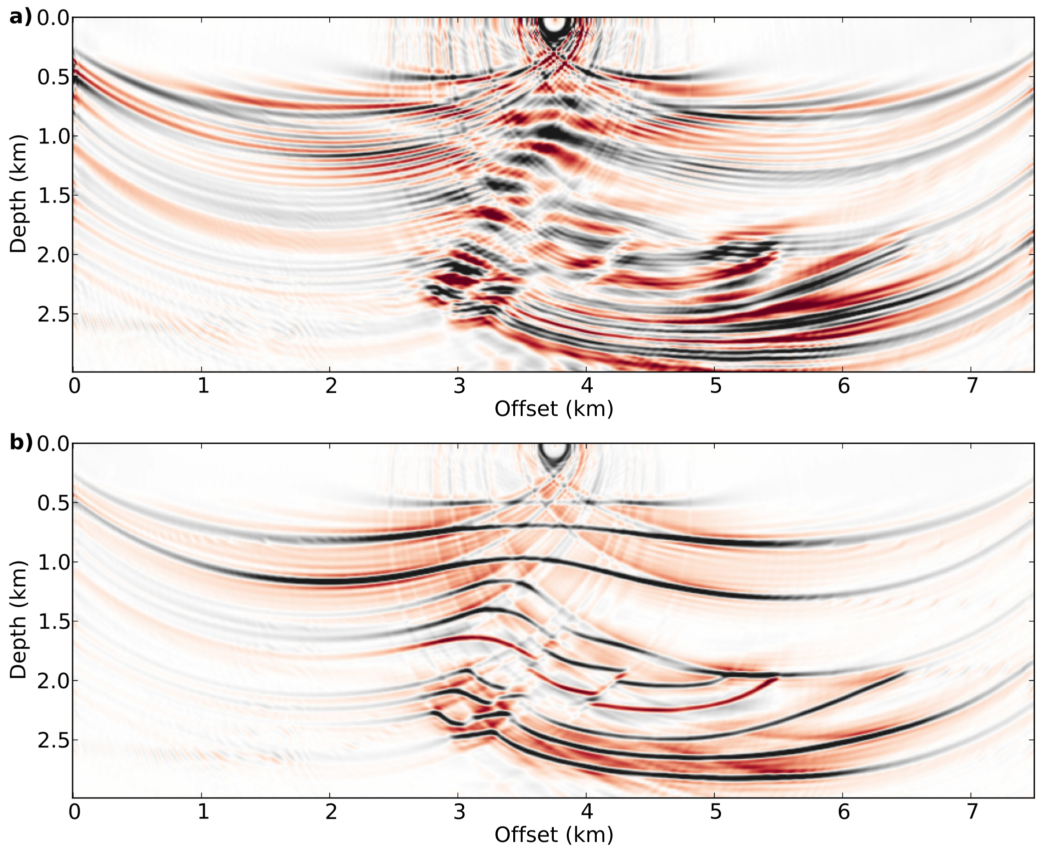

Figure 2.3: Least-squares migration for our first example. a) Image computed via the method of conjugate gradients and pseudo-adjoint operator. The solution corresponds to 14 iterations of the conjugate gradients method. (b) Image computed via the method of preconditioned conjugate gradients after 14 iterations and exact adjoint operator.

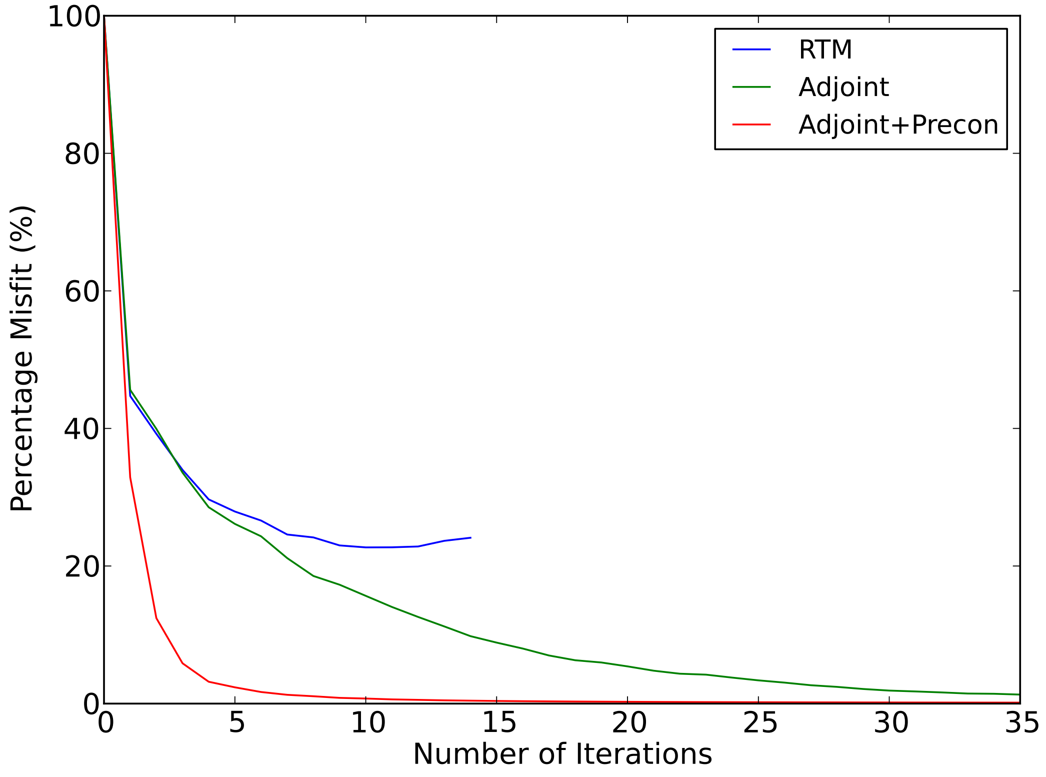

Figure 2.4: Converge curves for the conjugate gradients method for example 1. Blue curve: The pseudo-adjoint operator was adopted and the method starts to diverge at about 14 iterations. Green curve: Conjugate gradients with the exact adjoint operator. Red curve: Preconditioned conjugate gradients with exact adjoint operator.

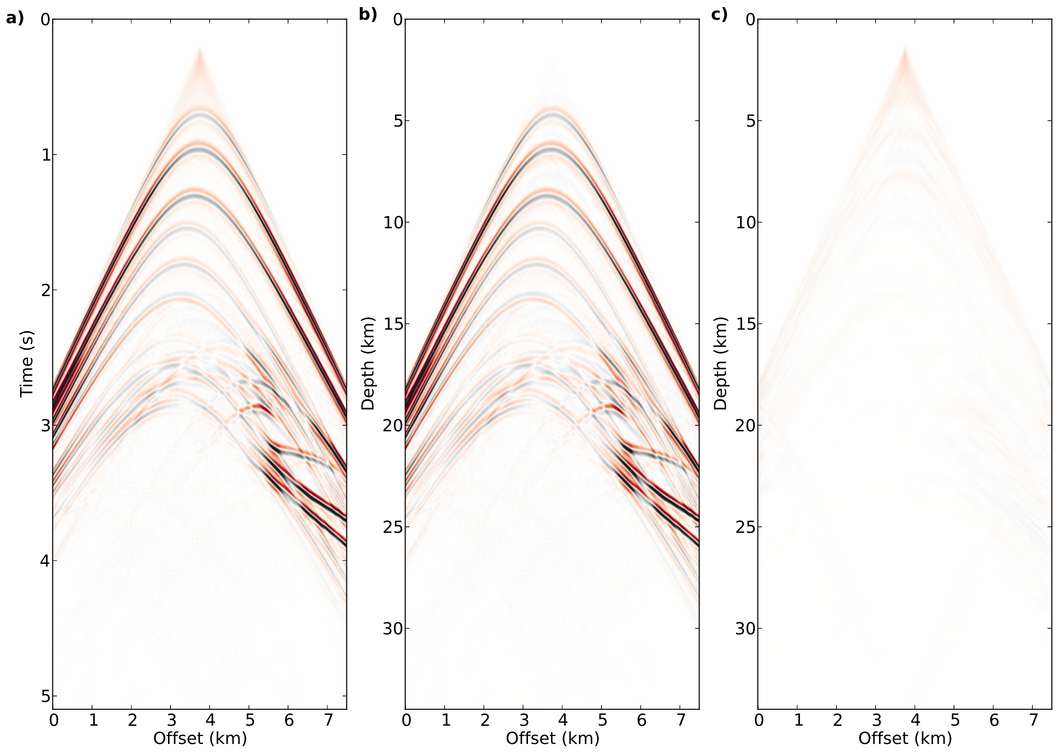

Figure 2.5: (a) Observed data for our first example . (b) Predicted data via preconditioned least-squares migration at 14 iterations. (c) Data residual.

2.6.2 Example 2: SAIG velocity model with multiple shots and inversion of an extended image

The second example is an extension of the previous one. From now on, the proposed least-squares two-way wave equation migration will only utilize the exact adjoint operator. In addition, all the examples have been computed with source-side energy preconditioning as described in the first example to normalize migration images (Valenciano and Biondi, 2003; Guitton et al., 2006). In this example, the synthetic survey includes 81 shots that are evenly distributed on the surface of the model. To accelerate the convergence rate of the proposed least squares migration, the least-squares migration is solved for extended shot-index images instead of one single image ($\mathbf{m}$) (Huang et al., 2016). In other words, rather than inverting for a single stacked image for all the shots (Figure 2.6a), the least-squares migration inverts for a image volume (Figure 2.6b) which, in this case, contains 81 partial images for the corresponding 81 shots. The image volume is indexed by shot number $\mathbf{m}_j$.

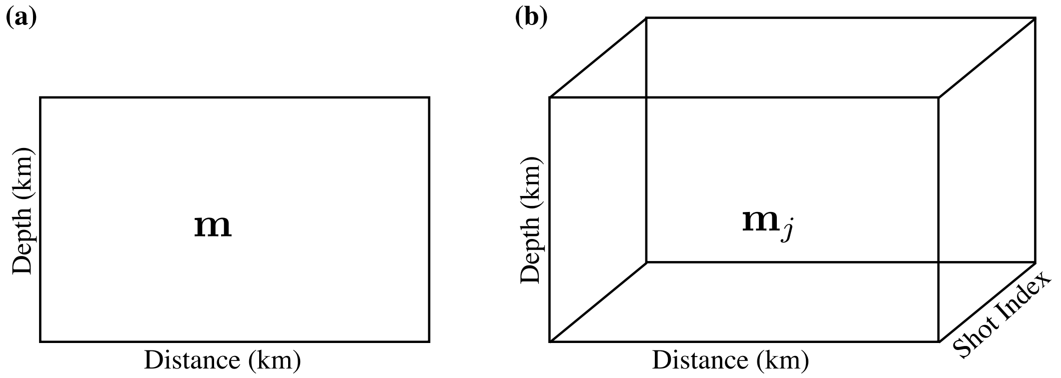

Figure 2.6: Solutions of acoustic least-squares two-way wave equation migration. (a) Stacked image. (b) Partial image volume indexed by shot number. Panel (b) shows the preferred solution for multi-shot surveys.

I also require lateral continuity across shots at any given subsurface point x, z. For this purpose, I define a cost function that is given by

$$J = \sum_{j=1}^{N_s} \|\mathbf{W}_{d_j}\,(\mathbf{L}_j \mathbf{P}_j (\frac{1}{2}\mathbf{a}_{j-1} + \mathbf{a}_j + \frac{1}{2}\mathbf{a}_{j+1}) - \mathbf{d}_j^{obs})\|_2^2 + \mu^2 \sum_{j=1}^{N_s} \|\mathbf{a}_j\|_2^2 \tag{2.31}$$

where

$$\mathbf{m}_j = \frac{1}{2}\mathbf{a}_{j-1} + \mathbf{a}_j + \frac{1}{2}\mathbf{a}_{j+1} \tag{2.32}$$

I now invert the auxiliary images $\mathbf{a}_j \quad j = 1, \dots N_s$ rather that inverting directly $\mathbf{m}_j \quad j = 1, \dots N_s$. In essence, I are simultaneously inverting $\mathbf{a}_j$ and forming the final individual images as a weighted average of nearby **a**-images (Wang et al., 2005). The approach serves to estimate extended images that are low-pass in the shot-index direction

$$m(x,z)_j = \frac{1}{2}\,a(x,z)_{j-1} + \,a(x,z)_j + \frac{1}{2}\,a(x,z)_{j+1}\,.$$

Figures 2.8 a and b show the migration of the 81 shots and the least-squares migration. In this case, the method of conjugate gradients was run until the misfit was reduced to about 1%. The latter happens at iteration 20 (shown in Figure 2.7). The images correspond to the average of the 81 partial images $\mathbf{m}_i$. To continue with our analysis, I examine the importance of smoothing across shot-index coordinates and its impact on the estimation of images for fix spatial position x. Figures 2.9a and b show the shot-index image computed for lateral position $x = 2.25$Km for the classical migration and for least-squares migration with preconditioning. Figures 2.10 and 2.11 show similar information for shot-index images computer at the centre and right side of the model ($x = 3.75$km and 5.25km), respectively.

Figures 2.9-2.11 show the importance of adding preconditioning in the form of lateral continuity constraints to minimize aperture and sampling artifacts in shot-index images. In this example, our averaging operator was a simple 3 points smoothing operator. I have tested our algorithm by varying the length of the smoothing operator. Long operators will produce overly smooth shot-index images at the cost of degrading data fidelity. For this simple velocity model, a 3-point smoothing operator provides a good trade-off between smoothness in shot-index images and misfit reduction. One shortcoming of least-squares migration is that one needs to explore trade-off curves for the scalar μ as well. In this work, I have preferred to fix $\mu = 0.001$ and use the number of iterations as a stopping criterion. For instance, the algorithm stops when the normalized misfit is below a predefined threshold. For this simple velocity model, the threshold is 1%. In essence, the number of iterations of the method of conjugate gradients is used to control the degree of fitting. In other words, the number of iteration is used as an effective trade-off parameter. The latter is rigorously discussed in Hansen (1987)'s book.

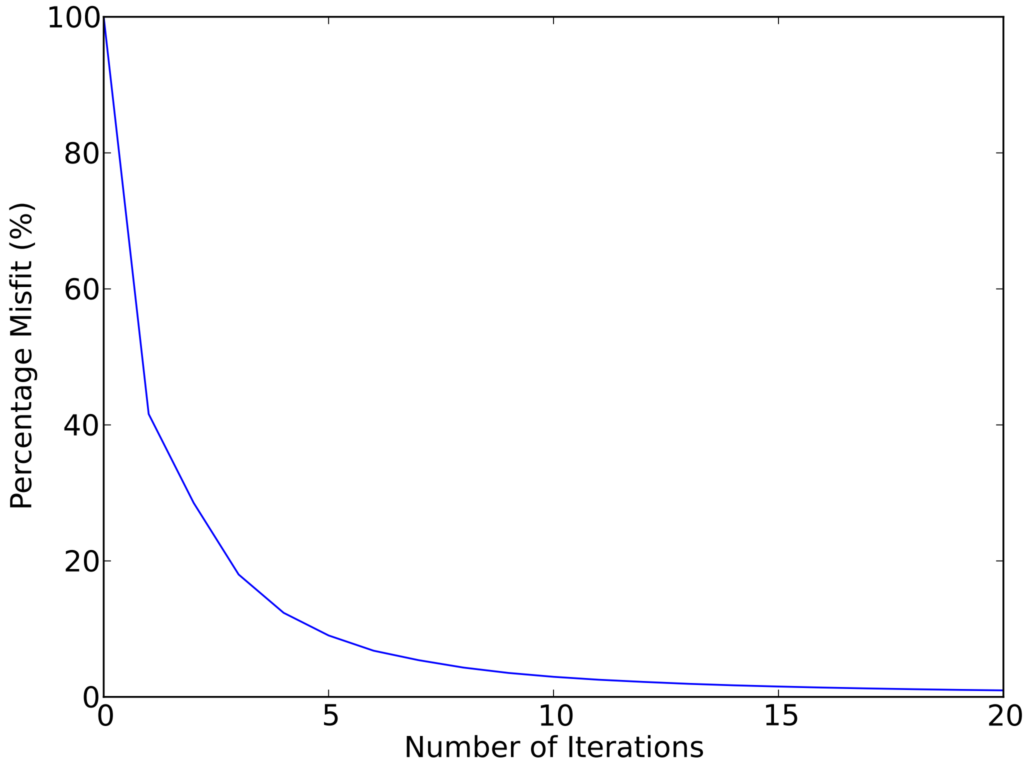

Figure 2.7: Convergence curves for the conjugate gradients method.

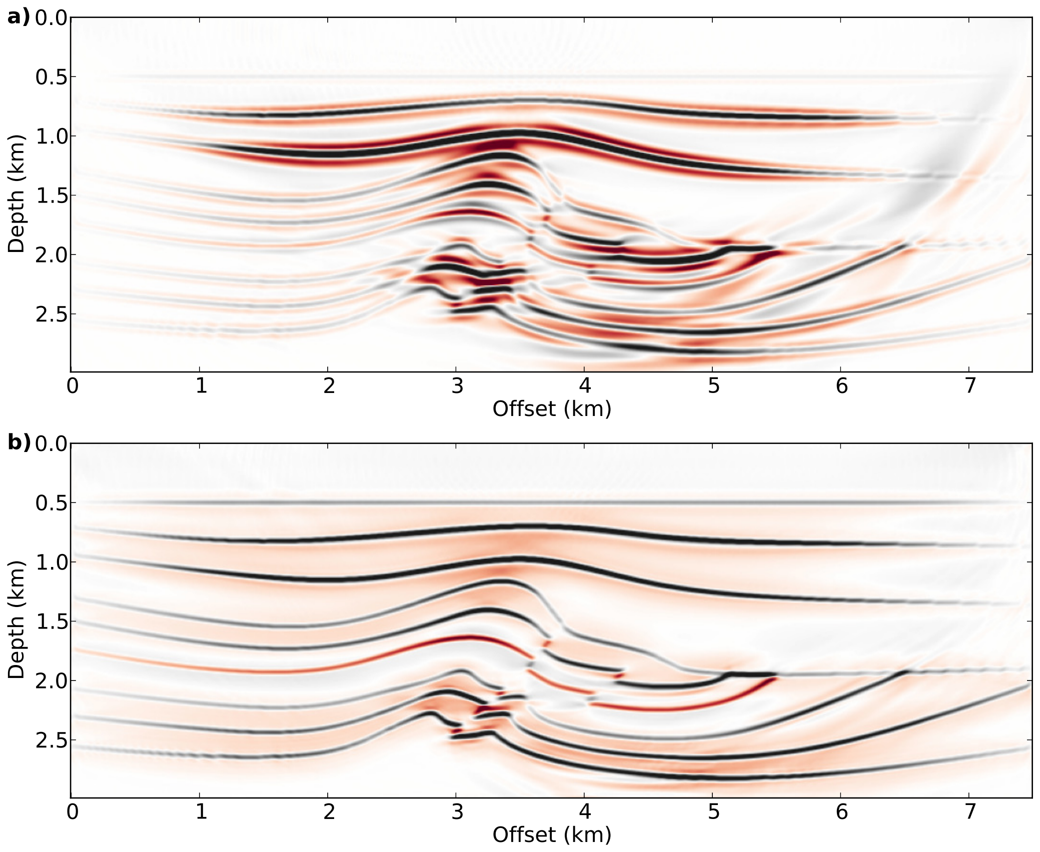

Figure 2.8: (a) Reverse time migration, sum of images computed from 81 shots gather. (b) Least-squares migration after 20 iterations where the 81 shots were simultaneously inverted with preconditioning to accentuate lateral continuity in shot-index images. Panel (b) displays the sum of the 81 inverted shot-index gathers.

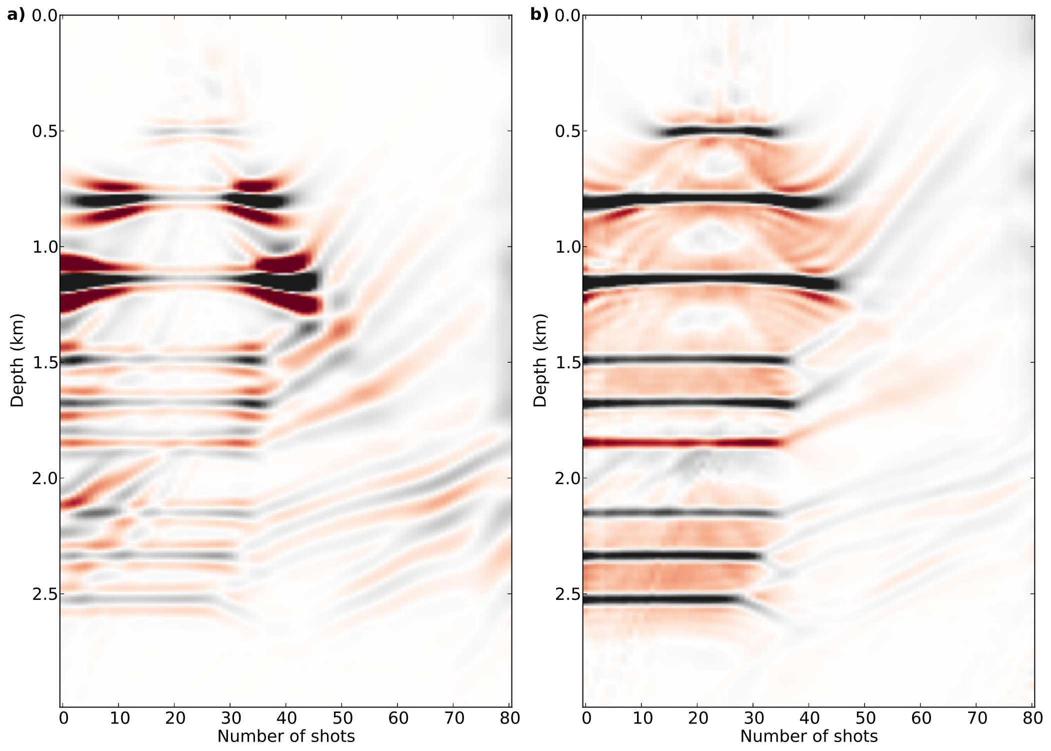

Figure 2.9: Shot-index image at position $x = 2.25$ km. (a) Migration. (b) Least-squares migration with preconditioning after 20 iterations.

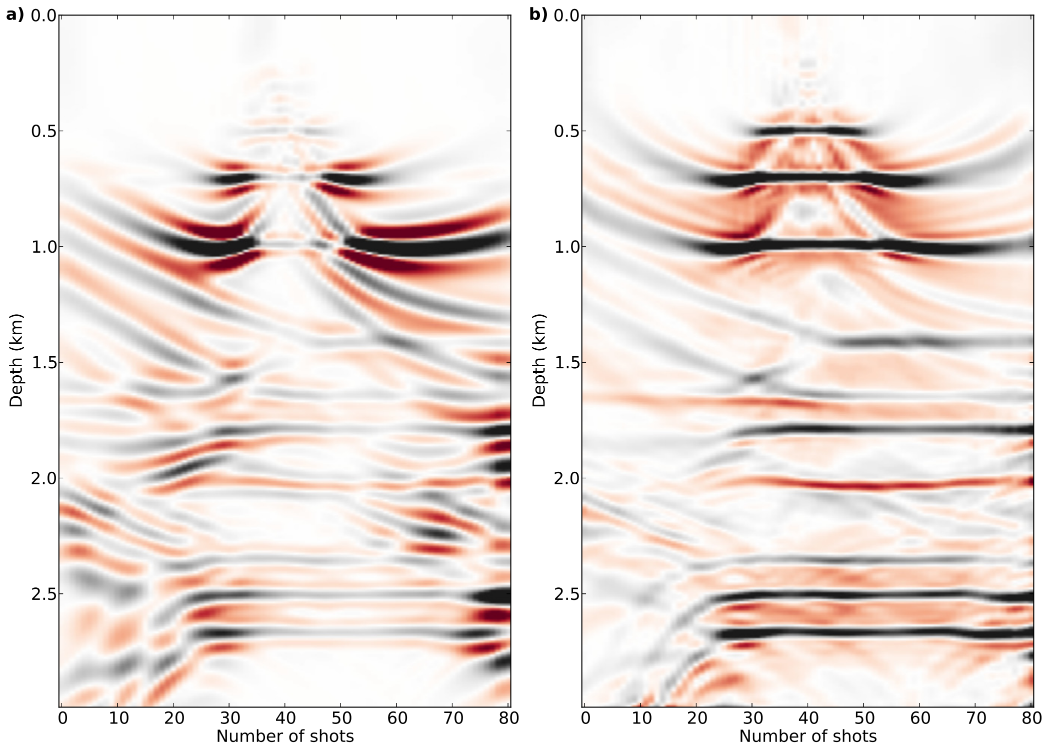

Figure 2.10: Shot-index image at position $x = 3.75$ km. (a) Migration. (b) Least-squares migration with preconditioning after 20 iterations.

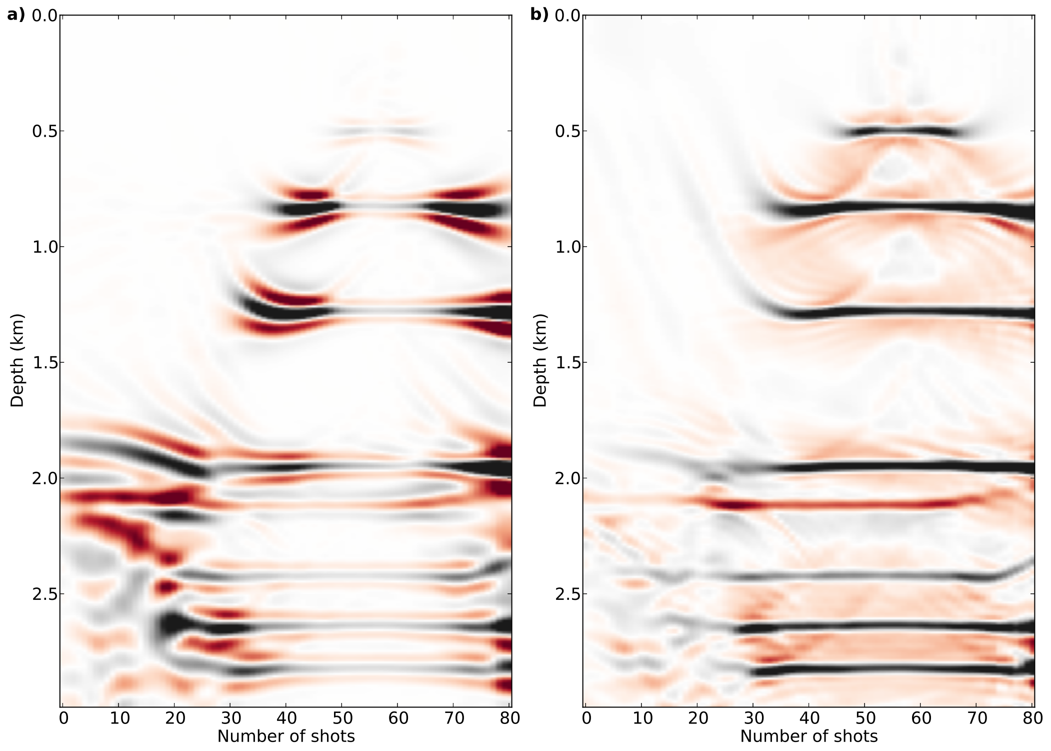

Figure 2.11: Shot-index image at position $x = 5.25$ km. (a) Migration. (b) Least-squares migration with preconditioning after 20 iterations.

2.6.3 Marmousi data set

In our last example, I use the Marmousi model (Figure 2.12) to test the performance of our least-squares migration algorithm. In this experiment, 81 shots are evenly distributed on the surface of the earth. I have adopted source-side energy preconditioning weights and a 7-point smoothing filter across the shot-index dimension with weights $[1/20, 3/10, 3/4, 1, 3/4, 3/10, 1/20]$. The weights correspond to applying the smoothing operator $[1/2, 1, 1/2]$ three times and normalization. After 35 iterations of least-squares migration, the data misfit is reduced to a predefined target of 10%. In examples 1 and 2 I were able to reduce the data misfit to about 1%. On the other hand, when working with the Marmousi model I have decided to stop the algorithm at about a 10% data reduction. This was needed to avoid generating artifacts in the extended image. The data set associated with the Marmousi model contains waves that are not modeled by the linearized forward operator. Therefore, one should not attempt to fit the data precisely. Figures 2.13a and b present our migration and least-squares migration results, respectively. These results correspond to stacks obtained by summing individually migrated or inverted shot-index images. The deblurring capability of least-squares migration (Zeng et al., 2014) is visible in Figure 2.14. I also computed the shot-index image for shot number 41. Figure 2.15 a and b illustrate the shot-index image obtained via migration and least-squares migration, respectively. Again, the addition of lateral smoothing across the shot-index coordinate has lead to a substantial attenuation of artifacts. Last, Figure 2.16 shows the prediction of shot 41 at iteration 35 of the conjugate gradients method. As it was already mentioned, the misfit for the 81 shots gathers was reduced to about 10%.

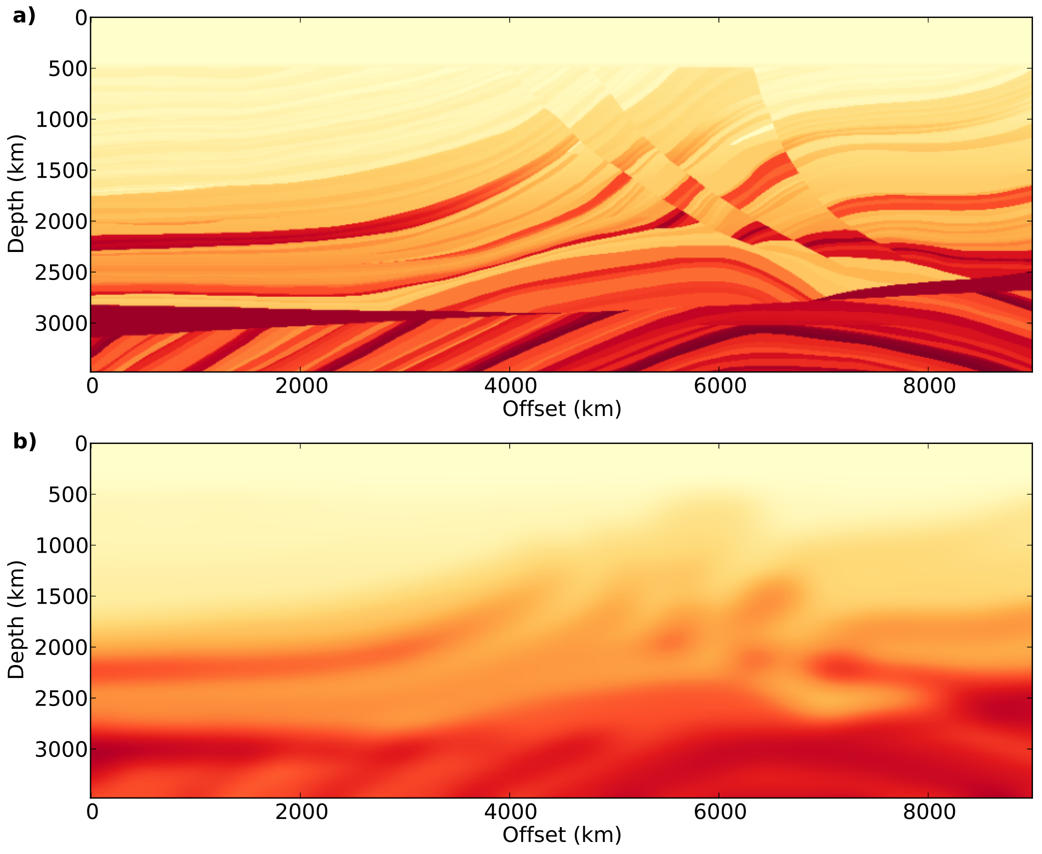

Figure 2.12: (a) True velocity for the Marmousi model. (b) Smooth velocity model.

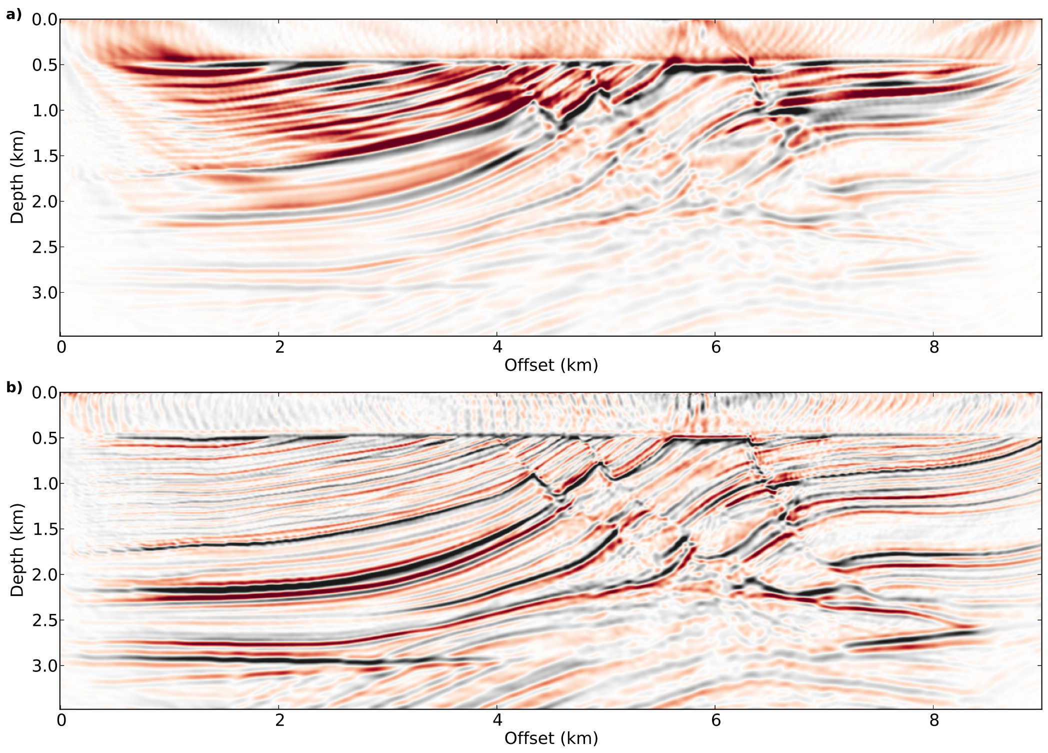

Figure 2.13: Migration of 81 shots. (a) Reverse time migration. (b) Least-squares migration after 35 iterations (preconditioning in shot-index images was applied)

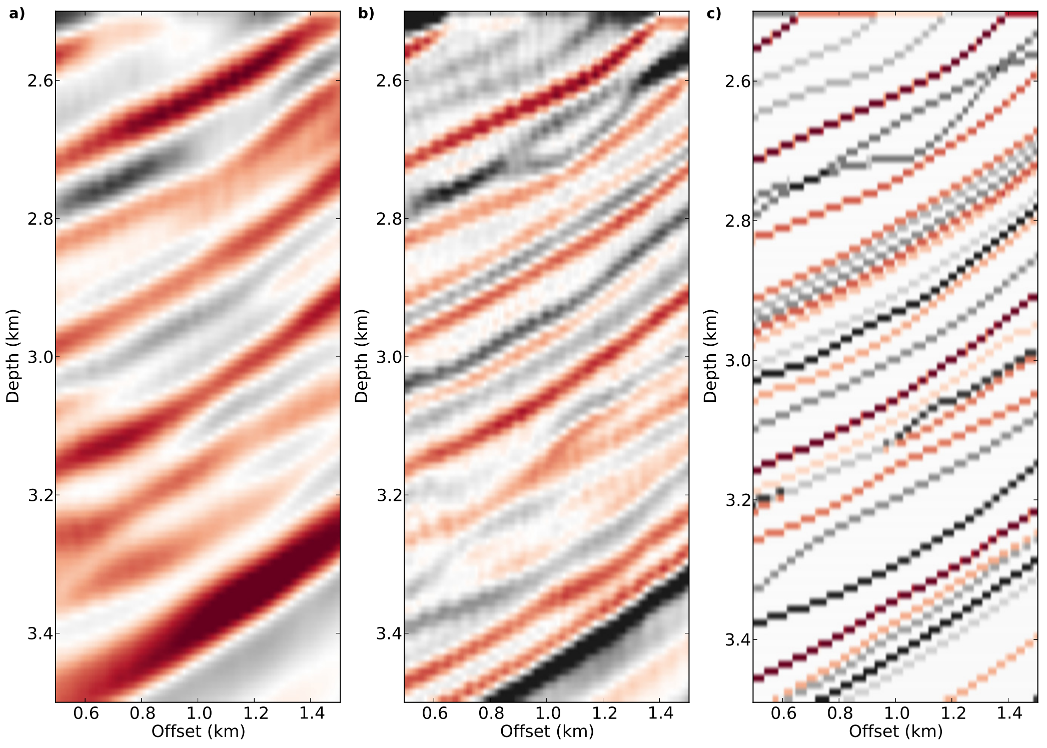

Figure 2.14: Details of the Marmousi model for x in the range 2.50-3.50 km and z in $0.50 - 1.50$km. (a) Reverse time migration. (b) Least-squares migration after 35 iterations. (c) Vertical derivative of the velocity model. The last image is used to represent reflectivity.

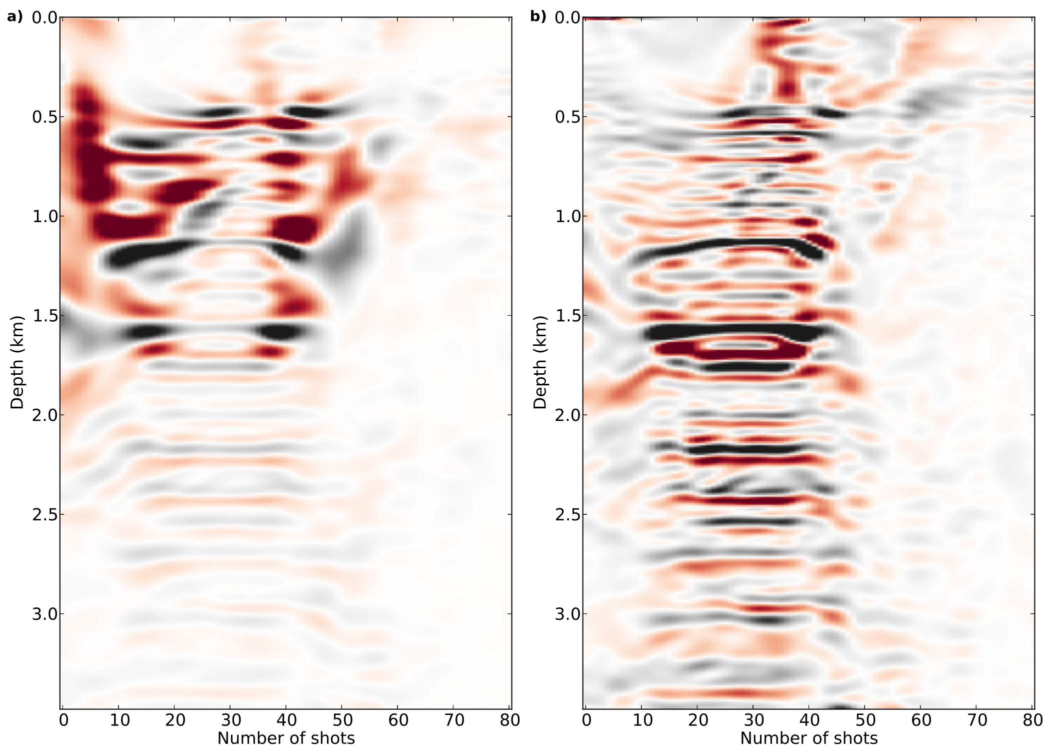

Figure 2.15: Shot-index image at $x = 4.00$ km of the Marmousi model. (a) Reverse time migration. (b) Least-squares migration reverse time after 35 iterations.

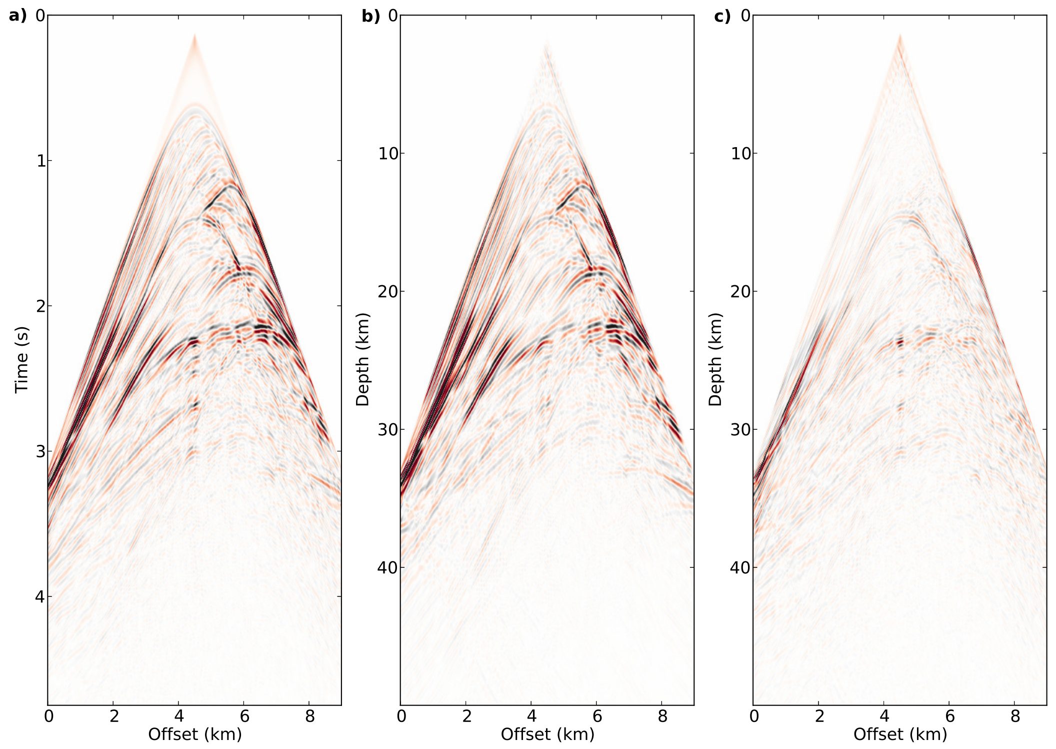

Figure 2.16: (a) The 41th shot record of the synthetic Marmousi dataset. (b) Shot prediction using the image inverted via least-squares migration. (c) Data residual.

CHAPTER 3

Elastic Least Squares Two-way Wave Equation Migration

3.1 Introduction

This chapter introduces elastic least-squares migration in terms of perturbations of Lame's parameters. Following Chapter 2, the Born elastic approximation is used to linearized the inverse problem. Green's function in smooth elastic media is computed via the method of finite difference. Once the forward linearized operator is derived, special attention is paid to designing the adjoint elastic operator. Finally, the method of conjugate gradients is used to estimate elastic images of the subsurface.

3.2 Forward modelling operator and its exact adjoint (multi-parameters)

Elastic wave propagation in a media with Lame parameters: $\lambda(x, z)$ and $\mu(x, z)$ and density $\rho(x, z)$ is governed the following set of partial differential equations:I

$$\begin{aligned}
\frac{\partial v_x}{\partial t} - \frac{1}{\rho}(\frac{\partial \tau_x}{\partial x} + \frac{\partial \tau_s}{\partial z}) &= 0,\\
\frac{\partial v_z}{\partial t} - \frac{1}{\rho}(\frac{\partial \tau_z}{\partial z} + \frac{\partial \tau_s}{\partial x}) &= 0,\\
\frac{\partial \tau_x}{\partial t} - \lambda(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}) - 2\mu\frac{\partial v_x}{\partial x} &= \delta(x-x_s)src,\\
\frac{\partial \tau_z}{\partial t} - \lambda(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}) - 2\mu\frac{\partial v_z}{\partial z} &= \delta(x-x_s)src,\\
\frac{\partial \tau_s}{\partial t} - \mu(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}) &= 0, \qquad (3.1)
\end{aligned}$$

where τ_x are τ_z are normal stresses in the x and z directions respectively. τ_s is a shear stress that is perpendicular with the two normal stresses. Horizontal and vertical particle velocities are denoted as v_x and v_z, respectively. To simulate pure explosive point source, source function src is injected to the two normal stresses, τ_x and τ_z. To linearize the elastic wave equation, I assume the media parameter consists of known background media and unknown perturbations such that $\rho = \rho_0 + \delta\rho$, $\lambda = \lambda_0 + \delta\lambda$ and $\mu = \mu_0 + \delta\mu$. Correspondingly, wavefield components can be described as linear expressions as: $v_x(x,z) = v_{x0}(x,z) + \delta v_x(x,z)$, $v_x(x,z) = v_{x0}(x,z) + \delta v_x(x,z)$, $v_x(x,z) = v_{x0}(x,z) + \delta v_x(x,z)$, $v_x(x,z) = v_{x0}(x,z) + \delta v_x(x,z)$ and $v_x(x,z) = v_{x0}(x,z) + \delta v_x(x,z)$. To substitute these linear combinations into the equation 3.1 and eliminate multiplication of small perturbations, I can obtain two sets of equations:

$$\begin{aligned}
\frac{\partial v_{x0}}{\partial t} - \frac{1}{\rho_0}(\frac{\partial \tau_{x0}}{\partial x} + \frac{\partial \tau_{s0}}{\partial z}) &= 0,\\
\frac{\partial v_{z0}}{\partial t} - \frac{1}{\rho_0}(\frac{\partial \tau_{z0}}{\partial z} + \frac{\partial \tau_{s0}}{\partial x}) &= 0,\\
\frac{\partial \tau_{x0}}{\partial t} - \lambda_0(\frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z}) - 2\mu_0\frac{\partial v_{x0}}{\partial x} &= \delta(x-x_s)src,\\
\frac{\partial \tau_{x0}}{\partial t} - \lambda_0(\frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z}) - 2\mu_0\frac{\partial v_{z0}}{\partial z} &= \delta(x-x_s)src,\\
\frac{\partial \tau_{s0}}{\partial t} - \mu_0(\frac{\partial v_{x0}}{\partial z} + \frac{\partial v_{z0}}{\partial x}) &= 0, \qquad (3.2)
\end{aligned}$$

and

$$
\begin{aligned}
\frac{\delta\partial v_x}{\partial t} - \frac{1}{\rho_0}(\frac{\partial\delta\tau_x}{\partial x} + \frac{\partial\delta\tau_s}{\partial z}) &= \frac{1}{\rho_0^2}(\frac{\partial\tau_{xx0}}{\partial x} + \frac{\partial\tau_{xz0}}{\partial z})\delta\rho, \\
\frac{\delta\partial v_z}{\partial t} - \frac{1}{\rho_0}(\frac{\partial\delta\tau_z}{\partial z} + \frac{\partial\delta\tau_s}{\partial x}) &= \frac{1}{\rho_0^2}(\frac{\partial\tau_{zz0}}{\partial z} + \frac{\partial\tau_{xz0}}{\partial x})\delta\rho, \\
\frac{\partial\delta\tau_x}{\partial t} - \lambda_0(\frac{\partial\delta v_x}{\partial x} + \frac{\partial\delta v_z}{\partial z}) - 2\mu_0\frac{\partial v_x}{\partial x} &= (\frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z})\delta\lambda + 2\frac{\partial v_{x0}}{\partial x}\delta\mu, \\
\frac{\partial\delta\tau_x}{\partial t} - \lambda_0(\frac{\partial\delta v_x}{\partial x} + \frac{\partial\delta v_z}{\partial z}) - 2\mu_0\frac{\partial v_z}{\partial z} &= (\frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z})\delta\lambda + 2\frac{\partial v_{z0}}{\partial z}\delta\mu, \\
\frac{\partial\delta\tau_s}{\partial t} - \mu_0(\frac{\partial\delta v_x}{\partial z} + \frac{\partial\delta v_z}{\partial x}) &= (\frac{\partial v_{x0}}{\partial z} + \frac{\partial v_{z0}}{\partial x})\delta\mu.
\end{aligned}
\tag{3.3}
$$

The background media in equation 3.2 can be obtained via tomographic inversion, migration velocity analysis or full waveform inversion (Tarantola, 1984). I will assume a known background in order to permits us to solve the equation 3.2 via finite difference to obtain background wavefield for a given point source function *src* at $x = x_s$. In migration jargon, the wavefield is so-called *source-side wavefield.* Equation 3.3 is similar with equation 3.2 but source input is given by subsurface response (right hand side of equation 3.3), which consist of unknown media perturbation $\delta\rho$, $\delta\lambda$ and $\delta\mu$ and the source-side wavefield. I can write the subsurface response in matrix form as

$$
\underbrace{\begin{pmatrix} r_{v_x} \\ r_{v_z} \\ r_{\tau_x} \\ r_{\tau_z} \\ r_{\tau_s} \end{pmatrix}}_{\mathbf{r}} = \underbrace{\begin{pmatrix} \frac{1}{\rho_0^2}(\frac{\partial\tau_{x0}}{\partial x} + \frac{\partial\tau_{s0}}{\partial z}) & 0 & 0 \\ \frac{1}{\rho_0^2}(\frac{\partial\tau_{z0}}{\partial z} + \frac{\partial\tau_{s0}}{\partial x}) & 0 & 0 \\ 0 & \frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z} & 2\frac{\partial v_{x0}}{\partial x} \\ 0 & \frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z} & 2\frac{\partial v_{z0}}{\partial z} \\ 0 & 0 & \frac{\partial v_{x0}}{\partial z} + \frac{\partial v_{z0}}{\partial x} \end{pmatrix}}_{\mathbf{K}} \underbrace{\begin{pmatrix} \delta\rho \\ \\ \delta\lambda \\ \\ \delta\mu \end{pmatrix}}_{\mathbf{m}},
\tag{3.4}
$$

where The combination of $\delta\rho$, $\delta\lambda$ and $\delta\mu$ are considered as model and denoted as $\mathbf{m}$. The operator K is known, which consists of the source-side wavefield. The left side the of equation 3.4 represents subsurface response induced by the model perturbation $\mathbf{m}$. The response is denoted as $\mathbf{r}_{[.]}$. Following the same strategy applied in the Chapter 2, the proposed elastic least-squares migration is limited to solve for media perturbation of $\delta\lambda$ and $\delta\mu$ in order to avoid the artifacts of multi-parameter crosstalk with respect to density and to reduce computational cost. To eliminate density, equation 3.4 becomes

$$
\underbrace{\begin{pmatrix} r_{\tau_x} \\ r_{\tau_z} \\ r_{\tau_s} \end{pmatrix}}_{\mathbf{r}} = \underbrace{\begin{pmatrix} \frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z} & 2\frac{\partial v_{x0}}{\partial x} \\ \frac{\partial v_{x0}}{\partial x} + \frac{\partial v_{z0}}{\partial z} & 2\frac{\partial v_{z0}}{\partial z} \\ 0 & \frac{\partial v_{x0}}{\partial z} + \frac{\partial v_{z0}}{\partial x} \end{pmatrix}}_{\mathbf{K}} \underbrace{\begin{pmatrix} \delta\lambda \\ \\ \delta\mu \end{pmatrix}}_{\mathbf{m}}.
\tag{3.5}
$$

Equation 3.5 is valid for each subsurface point x, z and all time t. After discretizing each wavefield component and each media perturbation by $nx \times nz \times nt$ and $nx \times nz$ points, equation 3.5 can be converted into a matrix formulation as

$$\mathbf{r} = \mathbf{KSm}, \tag{3.6}$$

where $\mathbf{m} \in R^{2\,nx\,nz \times 1}$ represents two unknowns perturbations: $\mathbf{m} = [\delta\boldsymbol{\lambda}, \delta\boldsymbol{\mu}]^T$. $\mathbf{K} \in R^{3\,nx\,nz\,nt \times 2\,nx\,nz\,nt}$ contains wavefield information with respect to both space and time. Clearly, a matrix $\mathbf{S} \in R^{3\,nx\,nz\,nt \times\, 2\,nx\,nz}$ is required to spray model in time. Finally, equation 3.6 produces the subsurface response $\mathbf{r} \in R^{3\,nx\,nz \times 1}$ which is taken as initial values in equation 3.3 to solve for the corresponding perturbative wavefield δv_x, δv_z, $\delta \tau_x$, $\delta \tau_z$ and $\delta \tau_s$. In the least-squares migration, they are also called scattering wavefield. Numerically, the wavefield can be solved by finite difference method. In a compact form, the relation (equation 3.3) between the response and the scattering wavefield can be translated into matrix form as following

$$\delta\mathbf{u} = \mathbf{Gr}, \tag{3.7}$$

where $\mathbf{G}$ is the matrix expression of elastic finite difference method (see Appendix A2) to solve equation 3.3. I will incorporate a sampling operator $\mathbf{R}$ to extract particle velocity at receiver locations. Then, the seismic observation vector $\mathbf{d}^{obs}$ can be expressed by means of a linear relationship in terms of the model $\mathbf{m}$ as

$$\begin{aligned} \mathbf{d}^{obs} &= \mathbf{R}\delta\mathbf{u}, \\ &= \mathbf{RGr}, \\ &= \mathbf{RGKSm}. \end{aligned} \tag{3.8}$$

In equation 3.8, the observed seismic data $\mathbf{d}^{obs}$ defines a linear relationship with respect to media perturbations $\mathbf{m}$. The linear relationship can be represented by a linear operator $\mathbf{L} = \mathbf{RGKS}$. Then equation 3.8 becomes

$$\mathbf{d}^{obs} = \mathbf{Lm}, \tag{3.9}$$

where $\mathbf{L}$ is so-called forward operator. Since the forward operator is in matrix form, its adjoint operator is the transpose of the forward operator

$$\begin{aligned} \hat{\mathbf{m}} &= \mathbf{L}^T\mathbf{d}^{obs}, \\ &= \mathbf{S}^T\mathbf{K}^T\mathbf{G}^T\mathbf{R}^T\mathbf{d}^{obs}, \end{aligned} \tag{3.10}$$

where $\mathbf{L}^T$ represents the adjoint operator. In addition, $\mathbf{R}^T$ is the adjoint of sampling and $\mathbf{G}^T$ is equivalent to propagate the data backwards in time. The matrix $\mathbf{K}^T$ converts the responses into media perturbations at each time step. Finally, $\mathbf{S}^T$ means to integrate the perturbations over time to produce a model $\hat{\mathbf{m}}$. If I substitute equation 3.9 into equation 3.10, I have

$$\hat{\mathbf{m}} = \mathbf{L}^T\mathbf{L}\mathbf{m} \ . \tag{3.11}$$

Equation 3.11 shows that the migration image $\hat{\mathbf{m}}$ is not equal to the unknown media perturbation $\mathbf{m}$. In order to obtain $\mathbf{m}$, I construct a optimization problem in the follow form

$$J = \|\mathbf{L}\mathbf{m} - \mathbf{d}^{obs}\|_2^2 + \mu^2\|\mathbf{m}\|_2^2 \, . \tag{3.12}$$

The solution of the last equation is the unknown media perturbation where $\mathbf{m} = (\mathbf{L}^T\mathbf{L} + \mu I)^{-1}\hat{\mathbf{m}}$, where μ is a small damping parameter to make $\mathbf{L}^T\mathbf{L}$ invertible. The optimization problem is solved by conjugate gradient method, which requires the forward $\mathbf{L}$ and $\mathbf{L}^T$ to be an exact adjoint pair. Therefore, I have to understand the precise structure of $\mathbf{G}$ in $\mathbf{L}$ in order to design its exact adjoint operator $\mathbf{L}^T$.

3.3 Matrix-based forward operator G and its exact adjoint

In this section, I apply finite difference method to solve for the forward simulation problem in equation 3.3. In order to adopt the perfectly matched layer (PML) method to absorb artificial boundary reflections (Berenger, 1996), I convert the partial differential equations in equation 3.3 into ordinary differential equations. The five wavefield components are virtually decomposed into horizontal and vertical terms. As a result, the entire set of equation 3.3 is expanded to fifteen equations as follow

$$\begin{cases} \frac{\partial \delta v_x^x}{\partial t} + \kappa_x \delta v_x^x = \frac{1}{\rho_0}\frac{\partial \delta \tau_x}{\partial x} + \frac{1}{2} r_{v_x}, \\ \frac{\partial \delta v_x^z}{\partial t} + \kappa_z \delta v_x^z = \frac{1}{\rho_0}\frac{\partial \delta \tau_s}{\partial z} + \frac{1}{2} r_{v_x}, \\ \delta v_x = \delta v_x^x + \delta v_x^z, \end{cases} \tag{3.13}$$

$$\begin{cases} \frac{\partial \delta v_z^x}{\partial t} + \kappa_x \delta v_z^x = \frac{1}{\rho_0}\frac{\partial \delta \tau_s}{\partial x} + \frac{1}{2} r_{v_z}, \\ \frac{\partial \delta v_z^z}{\partial t} + \kappa_z \delta v_z^z = \frac{1}{\rho_0}\frac{\partial \delta \tau_z}{\partial z} + \frac{1}{2} r_{v_z}, \\ \delta v_z = \delta v_z^x + \delta v_z^z, \end{cases} \tag{3.14}$$

$$\begin{cases} \frac{\partial \delta \tau_x^x}{\partial t} + \kappa_x \delta \tau_x^x = (\lambda_0 + 2\mu_0)\frac{\partial \delta v_x}{\partial x} + \frac{1}{2} r_{\tau_x}, \\ \frac{\partial \delta \tau_x^z}{\partial t} + \kappa_z \delta \tau_x^z = \lambda_0 \frac{\partial \delta v_z}{\partial z} + \frac{1}{2} r_{\tau_x}, \\ \delta \tau_x = \delta \tau_x^x + \delta \tau_x^z, \end{cases} \tag{3.15}$$

$$\begin{cases} \frac{\partial \delta \tau_z^x}{\partial t} + \kappa_x \delta \tau_z^x = \lambda_0 \frac{\partial \delta v_x}{\partial x} + \frac{1}{2} r_{\tau_z}, \\ \frac{\partial \delta \tau_z^z}{\partial t} + \kappa_z \delta \tau_z^z = (\lambda_0 + 2\mu_0)\frac{\partial \delta v_z}{\partial z} + \frac{1}{2} r_{\tau_z}, \\ \delta \tau_z = \delta \tau_z^x + \delta \tau_z^z, \end{cases} \tag{3.16}$$

$$\begin{cases} \frac{\partial \delta \tau_s^x}{\partial t} + \kappa_x \delta \tau_s^x = \mu_0 \frac{\partial \delta v_z}{\partial x} + \frac{1}{2} r_{\tau_s}, \\ \frac{\partial \delta \tau_s^z}{\partial t} + \kappa_z \delta \tau_s^z = \mu_0 \frac{\partial \delta v_x}{\partial z} + \frac{1}{2} r_{\tau_s}, \\ \delta \tau_s = \delta \tau_s^x + \delta \tau_s^z, \end{cases} \tag{3.17}$$

Equation 3.13 can be expressed in matrix form (See Appendix A2) as

$$\begin{aligned} \underbrace{\begin{bmatrix} \delta \mathbf{v}_x^x \\ \delta \mathbf{v}_x^z \\ \delta \mathbf{v}_x \end{bmatrix}^{n+1}}_{\delta \mathbf{u}_1^{n+1}} &= \underbrace{\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{I} & \mathbf{I} & \mathbf{0} \end{bmatrix}}_{\mathbf{M}_4^{u1}} \Bigg(\underbrace{\begin{bmatrix} \mathbf{A}_1^{v_x} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A}_2^{v_x} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{M}_3^{u1}} \underbrace{\begin{bmatrix} \delta \mathbf{v}_x^x \\ \delta \mathbf{v}_z^z \\ \delta \mathbf{v}_x \end{bmatrix}^{n}}_{\delta \mathbf{u}_1^{n}} + \cdots \\ &+ \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{B}_1^{v_x}\mathbf{D}_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{M}_2^{u1}} \underbrace{\begin{bmatrix} \delta \tau_x^x \\ \delta \tau_x^z \\ \delta \tau_x \end{bmatrix}^{n}}_{\delta \mathbf{u}_3^{n}} + \underbrace{\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{B}_2^{v_x}\mathbf{D}_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\mathbf{M}_1^{u1}} \underbrace{\begin{bmatrix} \delta \tau_s^x \\ \delta \tau_s^z \\ \delta \tau_s \end{bmatrix}^{n}}_{\delta \mathbf{u}_5^{n}} \Bigg) + \mathbf{r}_{v_x}^{n+1} \end{aligned} \tag{3.18}$$

where $\mathbf{A}_i$ and $\mathbf{B}_i$ are diagonal matrices (see Appendix A2) and $\mathbf{D}_i$ are first order spatial derivative which can be obtained by adopting Kronecker products, $\delta \mathbf{u}_1$, $\delta \mathbf{u}_3$ and $\delta \mathbf{u}_5$ stands for the combination of $\delta \mathbf{v}_x$, $\delta \boldsymbol{\tau}_x$, $\delta \boldsymbol{\tau}_s$ and their virtual subsets, respectively. Similarly, I can write equation 3.14 to 3.17 in form of matrices as

$$\begin{aligned}\underbrace{\begin{bmatrix}\delta\mathbf{v}_z^x\\ \delta\mathbf{v}_z^z\\ \delta\mathbf{v}_z\end{bmatrix}^{n+1}}_{\delta\mathbf{u}_2^{n+1}} &= \underbrace{\begin{bmatrix}\mathbf{I}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{I}&\mathbf{0}\\ \mathbf{I}&\mathbf{I}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_4^{u2}}\left(\underbrace{\begin{bmatrix}\mathbf{A}_1^{v_z}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{A}_2^{v_z}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_3^{u2}}\underbrace{\begin{bmatrix}\delta\mathbf{v}_z^x\\ \delta\mathbf{v}_z^x\\ \delta\mathbf{v}_z\end{bmatrix}^{n}}_{\delta\mathbf{u}_2^{n}} + \cdots\right.\\ &\left. + \underbrace{\begin{bmatrix}\mathbf{0}&\mathbf{0}&\mathbf{B}_1^{v_z}\mathbf{D}_1\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_2^{u2}}\underbrace{\begin{bmatrix}\delta\tau_s^x\\ \delta\tau_s^z\\ \delta\tau_s\end{bmatrix}^{n}}_{\delta\mathbf{u}_5^{n}} + \underbrace{\begin{bmatrix}\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{B}_2^{v_z}\mathbf{D}_1\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_1^{u2}}\underbrace{\begin{bmatrix}\delta\tau_z^x\\ \delta\tau_z^z\\ \delta\tau_z\end{bmatrix}^{n}}_{\delta\mathbf{u}_4^{n}}\right) + \mathbf{r}_{v_z}^{n+1}\end{aligned} \tag{3.19}$$

$$\begin{aligned}\underbrace{\begin{bmatrix}\delta\tau_x^x\\ \delta\tau_x^z\\ \delta\tau_x\end{bmatrix}^{n+1}}_{\delta\mathbf{u}_3^{n+1}} &= \underbrace{\begin{bmatrix}\mathbf{I}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{I}&\mathbf{0}\\ \mathbf{I}&\mathbf{I}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_4^{u3}}\left(\underbrace{\begin{bmatrix}\mathbf{A}_1^{\tau_x}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{A}_2^{\tau_x}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_3^{u3}}\underbrace{\begin{bmatrix}\delta\tau_s^x\\ \delta\tau_s^z\\ \delta\tau_s\end{bmatrix}^{n}}_{\delta\mathbf{u}_3^{n}} + \cdots\right.\\ &\left. + \underbrace{\begin{bmatrix}\mathbf{0}&\mathbf{0}&\mathbf{B}_1^{\tau_x}\mathbf{D}_1\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_2^{u3}}\underbrace{\begin{bmatrix}\delta\mathbf{v}_x^x\\ \delta\mathbf{v}_x^z\\ \delta\mathbf{v}_x\end{bmatrix}^{n}}_{\delta\mathbf{u}_1^{n}} + \underbrace{\begin{bmatrix}\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{B}_2^{\tau_x}\mathbf{D}_1\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_1^{u3}}\underbrace{\begin{bmatrix}\delta\mathbf{v}_z^x\\ \delta\mathbf{v}_z^z\\ \delta\mathbf{v}_z\end{bmatrix}^{n}}_{\delta\mathbf{u}_2^{n}}\right) + \mathbf{r}_{\tau_x}^{n+1}\end{aligned} \tag{3.20}$$

$$\begin{aligned}\underbrace{\begin{bmatrix}\delta\tau_z^x\\ \delta\tau_z^z\\ \delta\tau_z\end{bmatrix}^{n+1}}_{\delta\mathbf{u}_4^{n+1}} &= \underbrace{\begin{bmatrix}\mathbf{I}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{I}&\mathbf{0}\\ \mathbf{I}&\mathbf{I}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_4^{u4}}\left(\underbrace{\begin{bmatrix}\mathbf{A}_1^{\tau_z}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{A}_2^{\tau_z}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_3^{u4}}\underbrace{\begin{bmatrix}\delta\tau_s^x\\ \delta\tau_s^z\\ \delta\tau_s\end{bmatrix}^{n}}_{\delta\mathbf{u}_1^{n}} + \cdots\right.\\ &\left. + \underbrace{\begin{bmatrix}\mathbf{0}&\mathbf{0}&\mathbf{B}_1^{\tau_z}\mathbf{D}_1\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_2^{u4}}\underbrace{\begin{bmatrix}\delta\mathbf{v}_x^x\\ \delta\mathbf{v}_x^z\\ \delta\mathbf{v}_x\end{bmatrix}^{n}}_{\delta\mathbf{u}_1^{n}} + \underbrace{\begin{bmatrix}\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{B}_2^{\tau_z}\mathbf{D}_1\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix}}_{\mathbf{M}_1^{u4}}\underbrace{\begin{bmatrix}\delta\mathbf{v}_z^x\\ \delta\mathbf{v}_z^z\\ \delta\mathbf{v}_z\end{bmatrix}^{n}}_{\delta\mathbf{u}_2^{n}}\right) + \mathbf{r}_{\tau_z}^{n+1}\end{aligned} \tag{3.21}$$

$$\begin{aligned}\underbrace{\begin{bmatrix}\delta\tau_s^x\\ \delta\tau_s^z\\ \delta\tau_s\end{bmatrix}^{n+1}}_{\delta\mathbf{u}_5^{n+1}} &= \underbrace{\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{I} & \mathbf{I} & \mathbf{0}\end{bmatrix}}_{\mathbf{M}_4^{u5}}\left(\underbrace{\begin{bmatrix}\mathbf{A}_1^{\tau_s} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{A}_2^{\tau_s} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}}_{\mathbf{M}_3^{u5}}\underbrace{\begin{bmatrix}\delta\tau_s^x\\ \delta\tau_s^z\\ \delta\tau_s\end{bmatrix}^{n}}_{\delta\mathbf{u}_5^{n}} + \cdots\right.\\ &\left. + \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{B}_1^{\tau_s}\mathbf{D}_1\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}}_{\mathbf{M}_2^{u5}}\underbrace{\begin{bmatrix}\delta\mathbf{v}_z^x\\ \delta\mathbf{v}_z^z\\ \delta\mathbf{v}_z\end{bmatrix}^{n}}_{\delta\mathbf{u}_2^{n}} + \underbrace{\begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{B}_2^{\tau_s}\mathbf{D}_1\\ \mathbf{0} & \mathbf{0} & \mathbf{0}\end{bmatrix}}_{\mathbf{M}_1^{u5}}\underbrace{\begin{bmatrix}\delta\mathbf{v}_x^x\\ \delta\mathbf{v}_x^z\\ \delta\mathbf{v}_x\end{bmatrix}^{n}}_{\delta\mathbf{u}_1^{n}}\right) + \mathbf{r}_{\tau_s}^{n+1}\end{aligned} \tag{3.22}$$

The vectors $\delta\mathbf{v}$, $\delta\boldsymbol{\tau}_z$ and their subsets are denoted as $\delta\mathbf{u}_2$, $\delta\mathbf{u}_4$, respectively. In finite difference method, spatial extrapolation can be written in the following matrix form:

$$\delta\mathbf{s}^{n+1} = \mathbf{T}\delta\mathbf{s}^{n} + \mathbf{r}^{n+1} \tag{3.23}$$

where

$$\delta\mathbf{s}^{n+1} = \begin{bmatrix}\delta\mathbf{u}_1\\ \delta\mathbf{u}_2\\ \delta\mathbf{u}_3\\ \delta\mathbf{u}_4\\ \delta\mathbf{u}_5\end{bmatrix}^{n+1} \qquad \delta\mathbf{s}^{n} = \begin{bmatrix}\delta\mathbf{u}_1\\ \delta\mathbf{u}_2\\ \delta\mathbf{u}_3\\ \delta\mathbf{u}_4\\ \delta\mathbf{u}_5\end{bmatrix}^{n} \qquad \delta\mathbf{r}^{n+1} = \begin{bmatrix}\delta\mathbf{r}_{u1}\\ \delta\mathbf{r}_{u2}\\ \delta\mathbf{r}_{u3}\\ \delta\mathbf{r}_{u4}\\ \mathbf{u5}\end{bmatrix}^{n+1}$$

$$\begin{aligned}\mathbf{T} = &\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{M}_4^{u3} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{M}_4^{u4} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{M}_4^{u5}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{M}_2^{u3} & \mathbf{M}_1^{u3} & \mathbf{M}_3^{u3} & \mathbf{0} & \mathbf{0}\\ \mathbf{M}_2^{u4} & \mathbf{M}_1^{u4} & \mathbf{0} & \mathbf{M}_3^{u5} & \mathbf{0}\\ \mathbf{M}_1^{u5} & \mathbf{M}_2^{u5} & \mathbf{0} & \mathbf{0} & \mathbf{M}_3^{u5}\end{bmatrix}\cdots\\ &\begin{bmatrix}\mathbf{M}_4^{u1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{M}_4^{u2} & \mathbf{0} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{M}_3^{u1} & \mathbf{0} & \mathbf{M}_2^{u1} & \mathbf{0} & \mathbf{M}_1^{u1}\\ \mathbf{0} & \mathbf{M}_3^{u2} & \mathbf{0} & \mathbf{M}_2^{u2} & \mathbf{M}_3^{u2}\\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix}\end{aligned} \tag{3.24}$$

Symbol $\mathbf{s}$ represents a time snapshot. Considering each snapshot $\mathbf{s}_i$ consisting the five wavefield components and their subsets ($\mathbf{s}_i = [\delta\mathbf{u}_1, \delta\mathbf{u}_2, \delta\mathbf{u}_3, \delta\mathbf{u}_4, \delta\mathbf{u}_5]$), the time evolution of the entire finite difference method can be written as an iterative multiplication of time stepping matrices. For instance, if I consider only four steps, the series of stepping matrices can be written as follow

$$\begin{bmatrix}\delta\mathbf{s}^1\\ \delta\mathbf{s}^2\\ \delta\mathbf{s}^3\\ \delta\mathbf{s}^4\end{bmatrix} = \underbrace{\begin{bmatrix}\mathbf{I}&\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{I}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{I}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{T}&\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{I}&\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{I}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{T}&\mathbf{I}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{I}&\mathbf{0}&\mathbf{0}&\mathbf{0}\\ \mathbf{T}&\mathbf{I}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{I}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}&\mathbf{I}\end{bmatrix}}_{\mathbf{G}}\begin{bmatrix}\mathbf{r}^1\\ \mathbf{r}^2\\ \mathbf{r}^3\\ \mathbf{r}^4\end{bmatrix}. \tag{3.25}$$

So far $\mathbf{G}$ (in equation 3.8) is fully constructed in a matrix formulation to propagates subsurface responses for all $\delta\mathbf{s}^n$ from $n = 1 \ldots nt$ (where $nt = 4$). The exact adjoint of $\mathbf{G}$ is the transpose of equation 3.25 and its sub-matrices. Certainly, it is not practical to save those gigantic matrices in the computer memory. Instead, I treat the $\mathbf{G}$ and $\mathbf{G}^T$ as operators (function in C language) and use loops to implement matrix multiplication (for the non-zeros entries only). Moreover, I note that the procedure leads to $\mathbf{G}$ and $\mathbf{G}^T$ (the forward operator 3.9 and its adjoint operator 3.10) that pass the dot-product test within machine precision.

3.4 Numerical examples

In this section, I use SAIG anticline model to demonstrate advantages of the elastic LS-RTM.The model has a grid size of 300×750 and spatial sampling of 7m in x and z as shown in Figure 3.1. The compressional velocity ranges from 1.5km/s to 3.0km/s. The shear velocity model is a scaled-down version of the previous model by $\sqrt{3}$. The migration velocity models are smoothed by a hamming filter in size of 75×75 as shown in Figure 3.2.

3.4.1 SAIG velocity model: single shot test

A single shot is placed in the center of the model (at 2.67 km) to generate observed data. Its direct wave is removed by an diagonal matrix $\mathbf{W}_d$ and migrated by the exact adjoint operator $\mathbf{L}^T$. The amplitude of the migrated image is balanced by a source-side illumination compensation operator, $\mathbf{P}$, (Valenciano and Biondi, 2003) where

$$P(ix, iz) = \tau_x^2(ix, iz) + \tau_z^2(ix, iz). \tag{3.26}$$

I also applied a vertical Laplacian filter (marked as $\mathbf{D}_v$) (Guitton et al., 2007) to remove near-surface low wavenumber artifacts on the final image.

$$\mathbf{m}_{migration} = \mathbf{D}_v\,\mathbf{P}\,\mathbf{L}^T\,\mathbf{W}_d\,\mathbf{d} \tag{3.27}$$

Figure 3.4 shows the adjoint image. In seismic imaging, the adjoint operator (equation 3.10) is also called migration. Compared Figure 3.4 with conventional elastic migration (Rosales and Rickett, 2001; Rosales et al., 2008; Yan and Sava, 2008; Lu et al., 2010; Du et al., 2012), the proposed adjoint operator does not require polarity correction because the adjoint operator attempts to solve for media perturbation. Physically, media perturbation does not have polarity reversal problem. However, Figure 3.4 is contaminated by cross-talk artifacts, which appears as fake layers between true reflectors. Next, I adopt the forward and its exact adjoint operator to derive least-squares image via solving the following equation.

$$J = \|\mathbf{W}_d(\mathbf{LPu} - \mathbf{d}^{obs})\|_2^2 + \mu\|\mathbf{u}\|_2^2 \tag{3.28}$$

where

$$\mathbf{m}_{LSM} = \mathbf{D}_v\mathbf{Pu} \tag{3.29}$$

The trade-off parameter μ is 10^{-3}. After applying 35 iterations of elastic LSRTM via conjugate gradients, the norm of the data residual is reduced to 1% (in Figure 3.3). The corresponding least squares image is shown in Figure 3.5. The elastic LSRTM removes the artificial layers, near surface layers are refined, amplitude of the deepest structures is enhanced and the final images are able to honour the data.

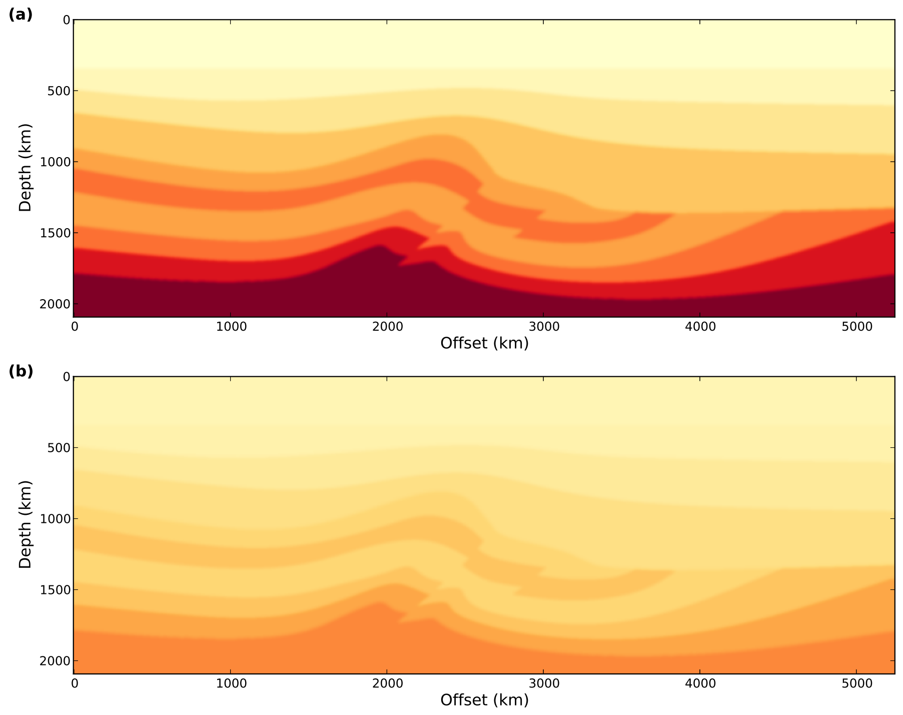

Figure 3.1: True velocity model. (a) Compressional velocity model. (b) Shear velocity model

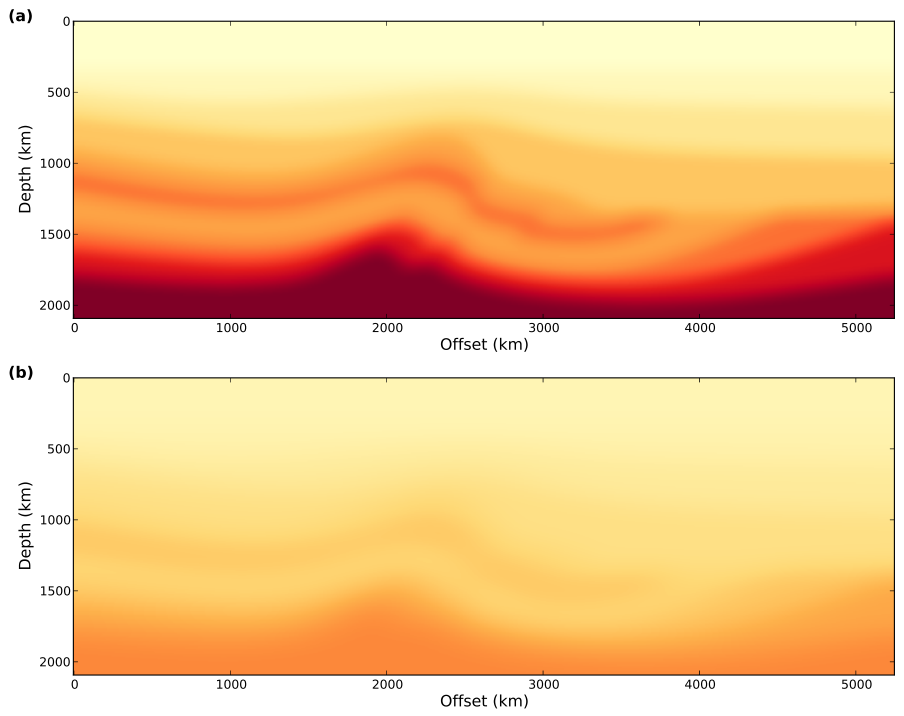

Figure 3.2: Smooth velocity model used to test least-squares migration. (a) Smooth compressional velocity model. (b) Smooth shear velocity model.

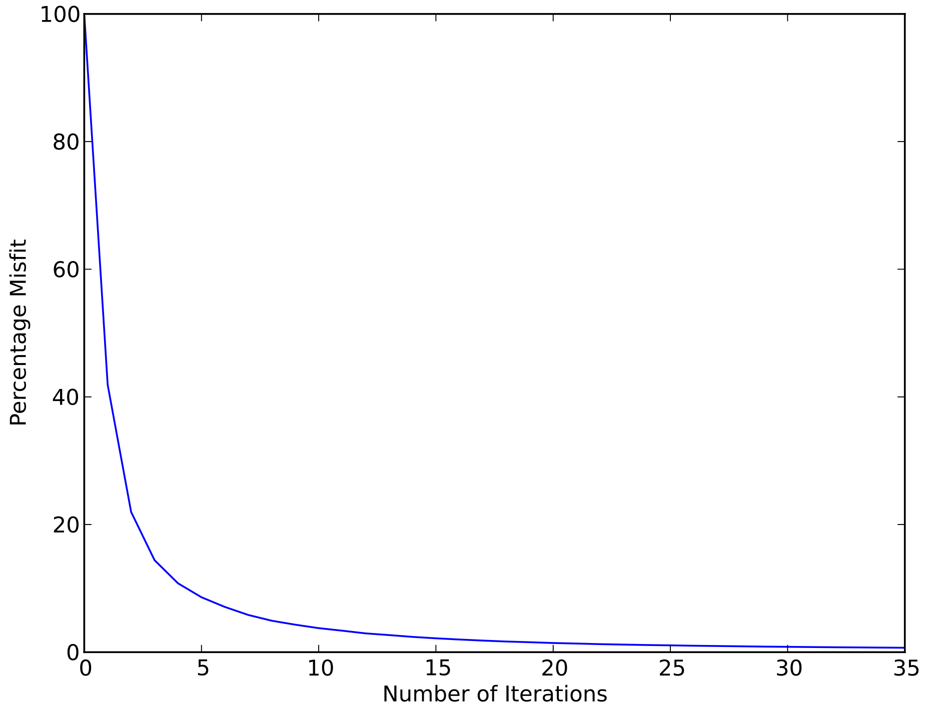

Figure 3.3: Convergence curves for the conjugate gradients method.

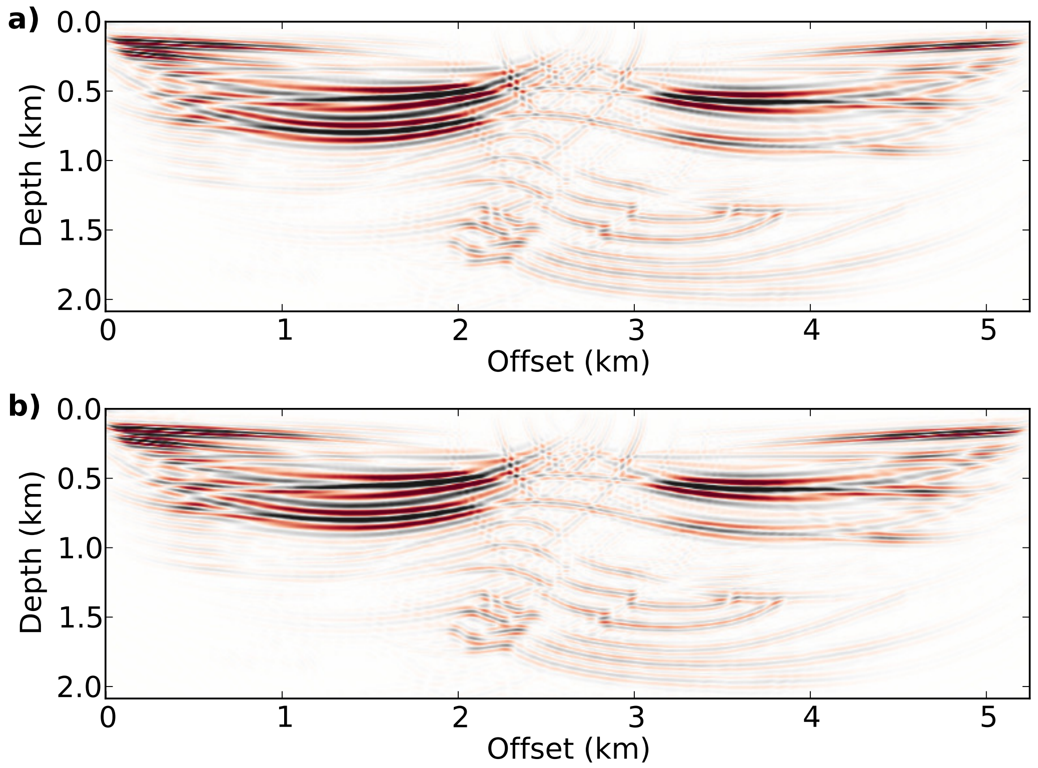

Figure 3.4: Migration of one shot gather. (a) Migrated perturbation of $\delta\lambda$. (b) Migrated perturbation of $\delta\mu$.

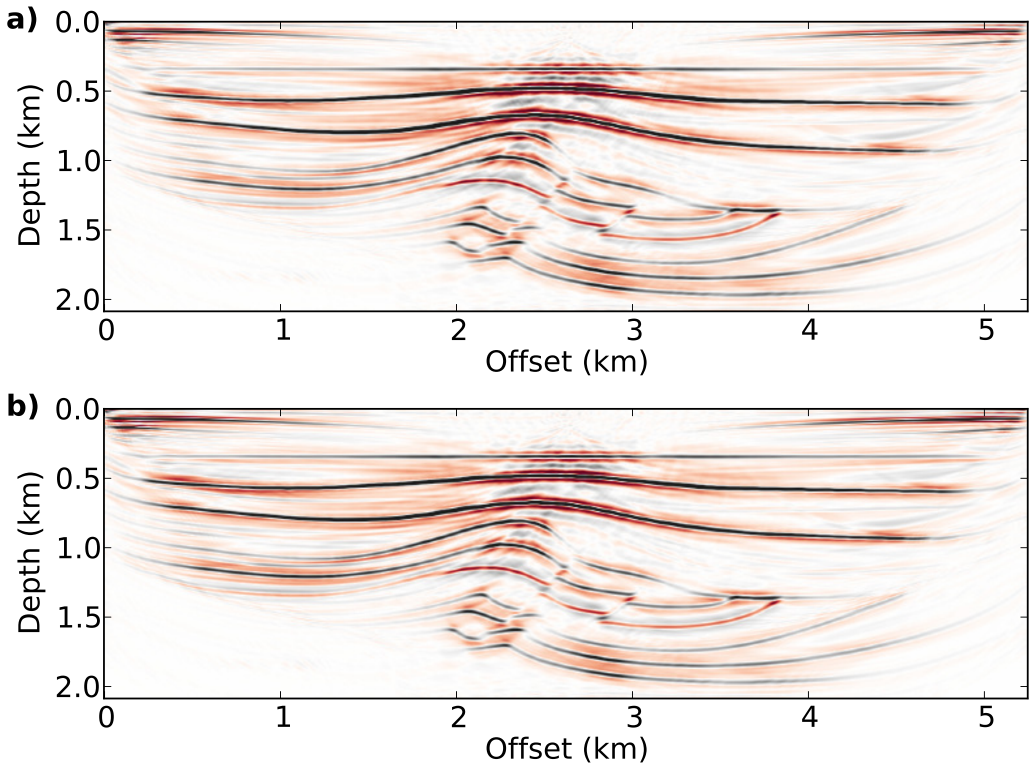

Figure 3.5: Least-squares migration of one shot gather corresponds to 35 iterations of the conjugate gradients method. (a) Inverted perturbation of $\delta\lambda$. (b) Inverted perturbation of $\delta\mu$.

3.4.2 SAIG velocity model: multiple shots test with smoothing in the shot-index coordinate

In this section, the previous study is extended to 81 shots, which are evenly distributed on the surface. The final image in the previous section has strong near surface artifact between 0 to 500 m in depth. Therefore, a diagonal matrix ($\mathbf{M}$) is included to remove near surface artifacts in the model domain. To accelerate the convergence rate, the proposed least-squares migration aims to solve for two image volumes of compressional and shear modulus with respect to each shot (Figure 3.6), rather than solving for two stacked images for all shots. The image volumes has higher degree of freedom to fit the observed data (Huang et al., 2016) but it also provides redundancy in the auxiliary shot-index axis to host artifacts. The artifacts appears as high frequency noise along the shot-index axis. In this case, a 3-point smoothing filter (equation 2.32) is implemented in form of matrix (denoted as

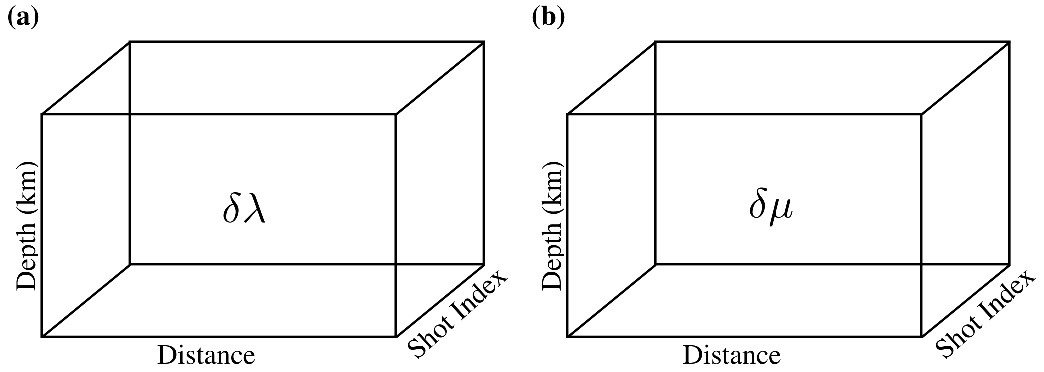

Figure 3.6: Partial image volumes indexed by shot numbers. (a) Image volume of $\delta\lambda$. (b) Image volume of $\delta\mu$.

E) to penalize large variation along shot axis (Wang et al., 2005). It is a similar procedure in acoustic least-squares migration (equation 2.32). After incorporating the matrices, **M** and **S**, migration operator is presented in the following equation

$$\mathbf{m} = \mathbf{M}\,\mathbf{E}\,\mathbf{P}\,\mathbf{L}^T\,\mathbf{W}_d\,\mathbf{d} \tag{3.30}$$

In this case, a vertical Laplacian filter is applied to the migration image for mitigating low-wavenumber artifacts. The final migration images are shown in figure 3.7. In both $\delta\lambda$ and $\delta\mu$ models, the near surface area shows strong amplitude artifact (at about 0.75 km in depth). In the $\delta\lambda$ models, deep layers are blurred by strong side-lobe artifacts (Zeng et al., 2014). The $\delta\mu$ model shows a slightly better resolution and has less side-lobes. However, the deep structure is still poorly illuminated. Next, I use both forward and adjoint operator to solve the following inversion problem.

$$J = \|\mathbf{W}_d(\mathbf{LSPMu} - \mathbf{d}^{obs})\|_2^2 + \mu\|\mathbf{u}\|_2^2 \tag{3.31}$$

where

$$\mathbf{m} = \mathbf{MSPu}. \tag{3.32}$$

After 35 iterations, LSRTM produces models which can predict about 95% of data (as shown in Figure 3.9). In Figure 3.8, the near surface layers are clearly separated. In the deep structures and side-lobes are suppressed. To analyze the upgraded image in details, I demonstrate three common image gathers located on left, center and right of the model. In Figure 3.10, 3.12, and 3.14, migration results have imbalanced amplitude with respect to shot locations. In the far offset, migration gathers have strong side-lobe problem. After 35 iterations, LSRTM produces the results showing better continuity along shot axis (in Figure

3.11, 3.13 and 3.15). After the side-lobes are suppressed, each layer is resolved. Eventually, the final image shows higher resolution structure than migration image.

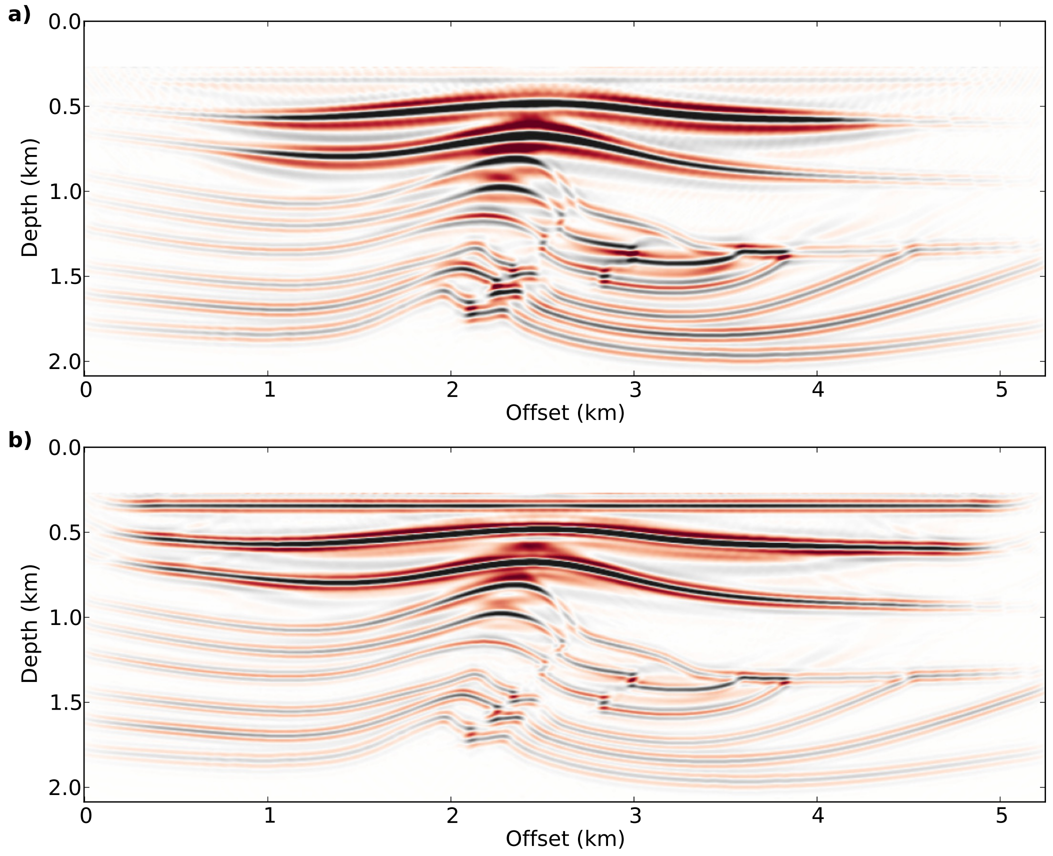

Figure 3.7: Migration of 81 shots. (a) Migrated image of $\delta\lambda$. (b) Migrated image of $\delta\mu$.

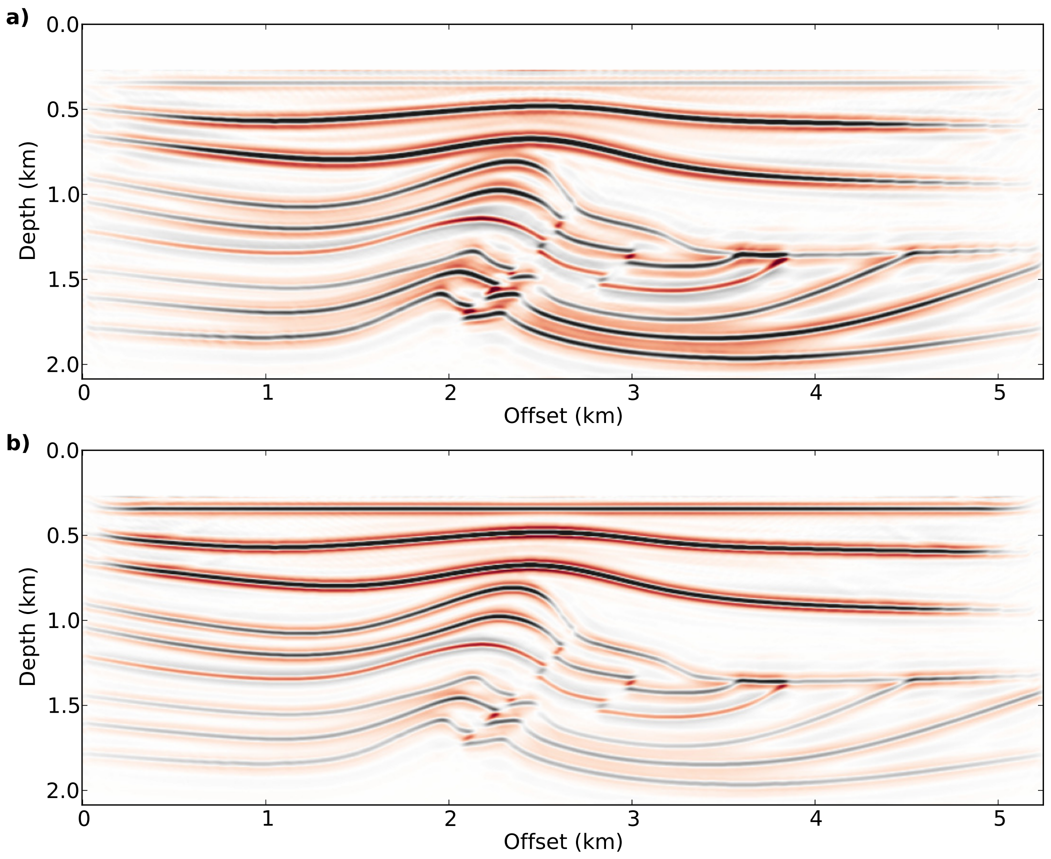

Figure 3.8: Least-squares migration after 35 iterations (preconditioning in shot-index images was applied). (a) Inverted image of $\delta\lambda$. (b) Inverted image of $\delta\mu$.

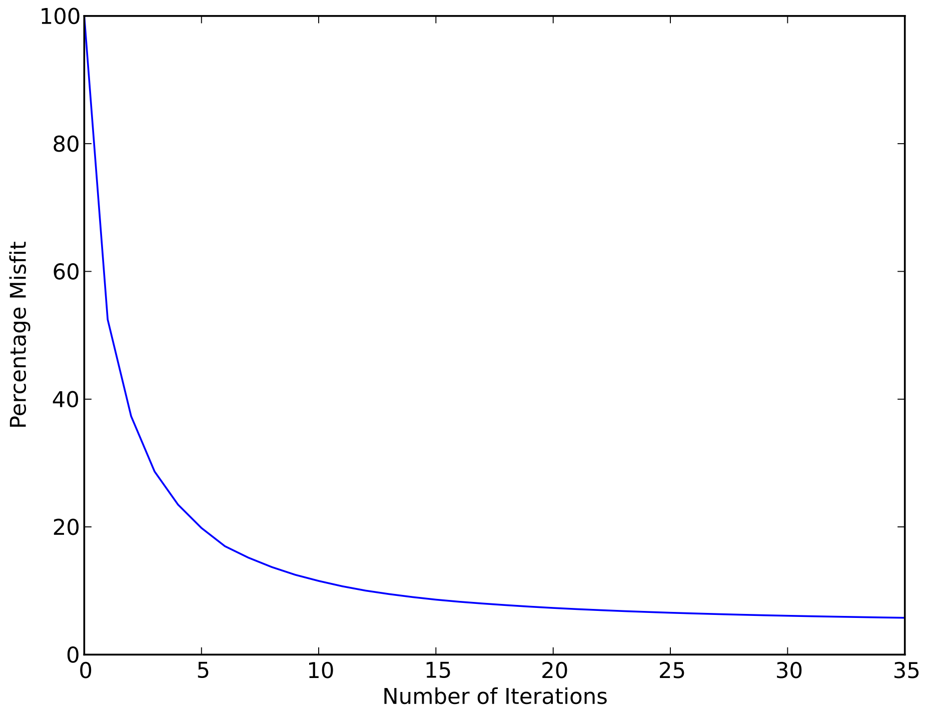

Figure 3.9: Convergence curves for the conjugate gradients method.

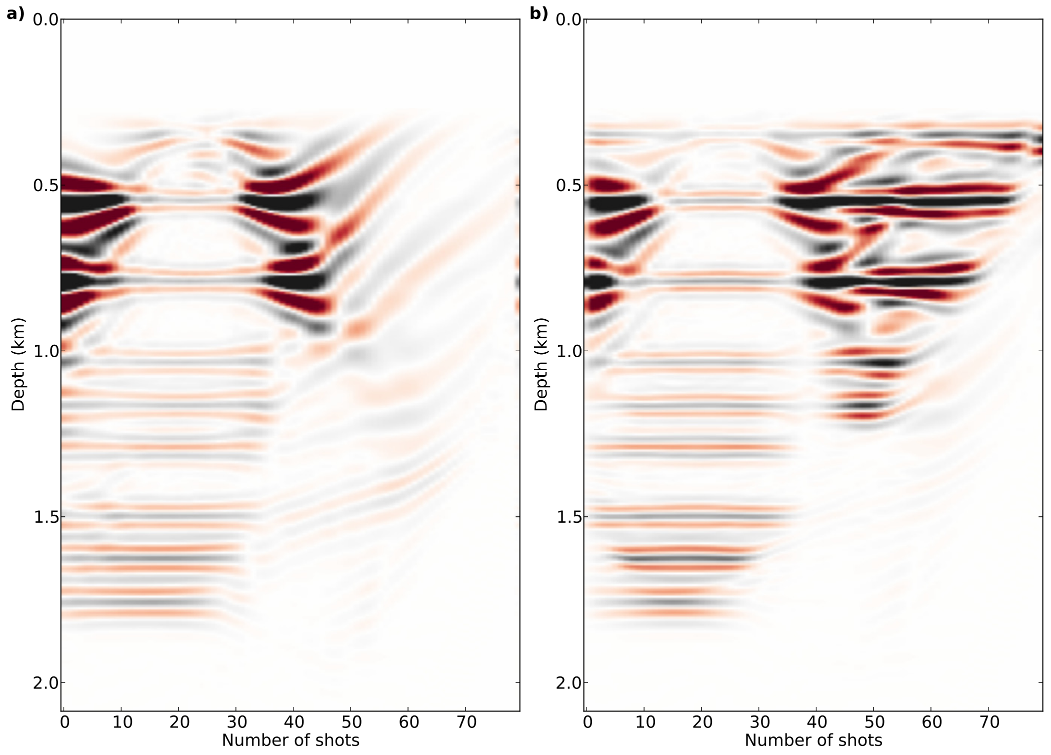

Figure 3.10: Shot-index migration image at $x = 2.25$ km. (a) Shot-index image of $\delta\lambda$. (b) Shot-index image of $\delta\mu$.

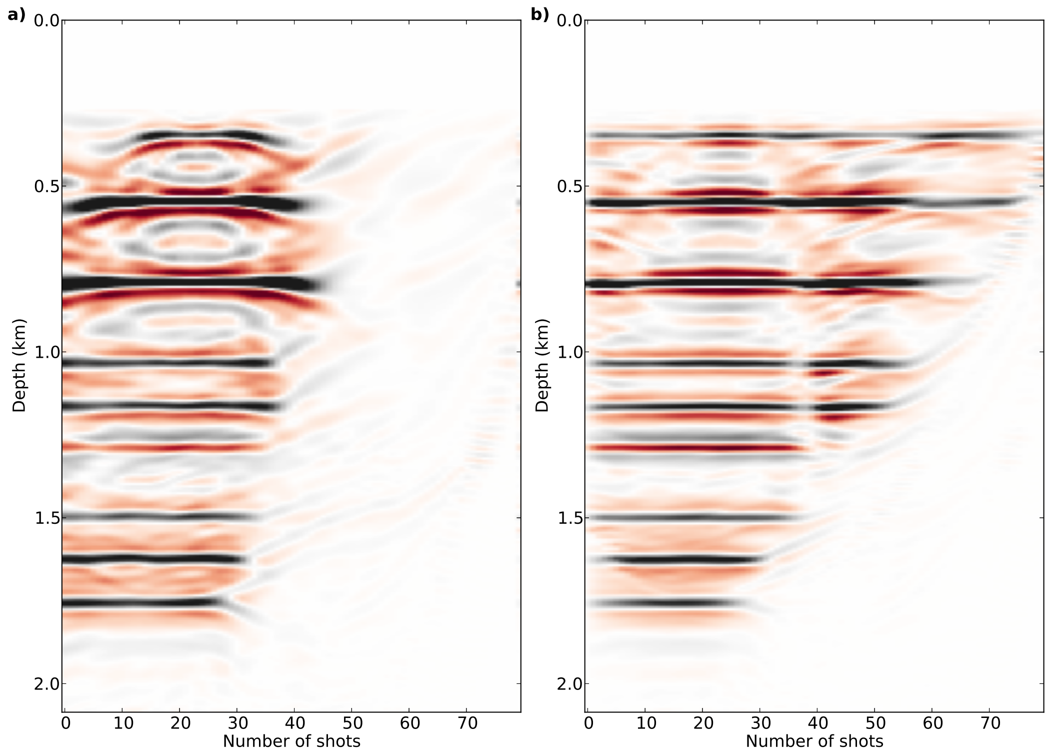

Figure 3.11: Shot-index least-squares migration image at $x = 2.25$ km. (a) Shot-index image of $\delta\lambda$. (b) Shot-index image of $\delta\mu$.

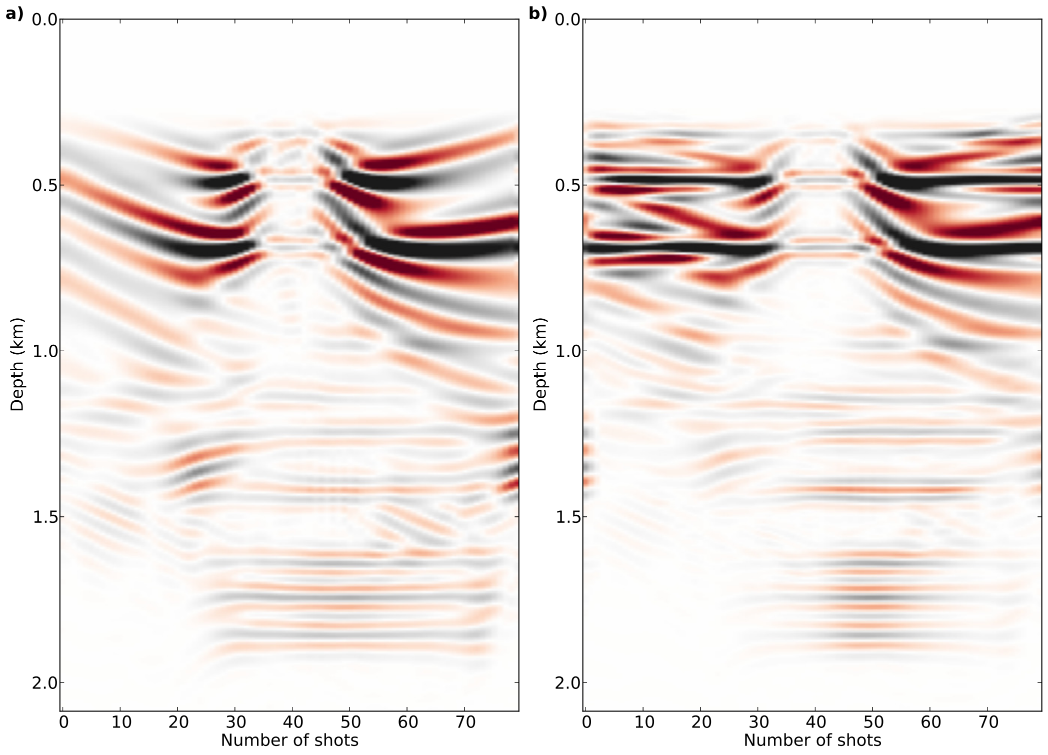

Figure 3.12: Shot-index migration image at $x = 3.75$ km. (a) Shot-index image of $\delta\lambda$. (b) Shot-index image of $\delta\mu$.

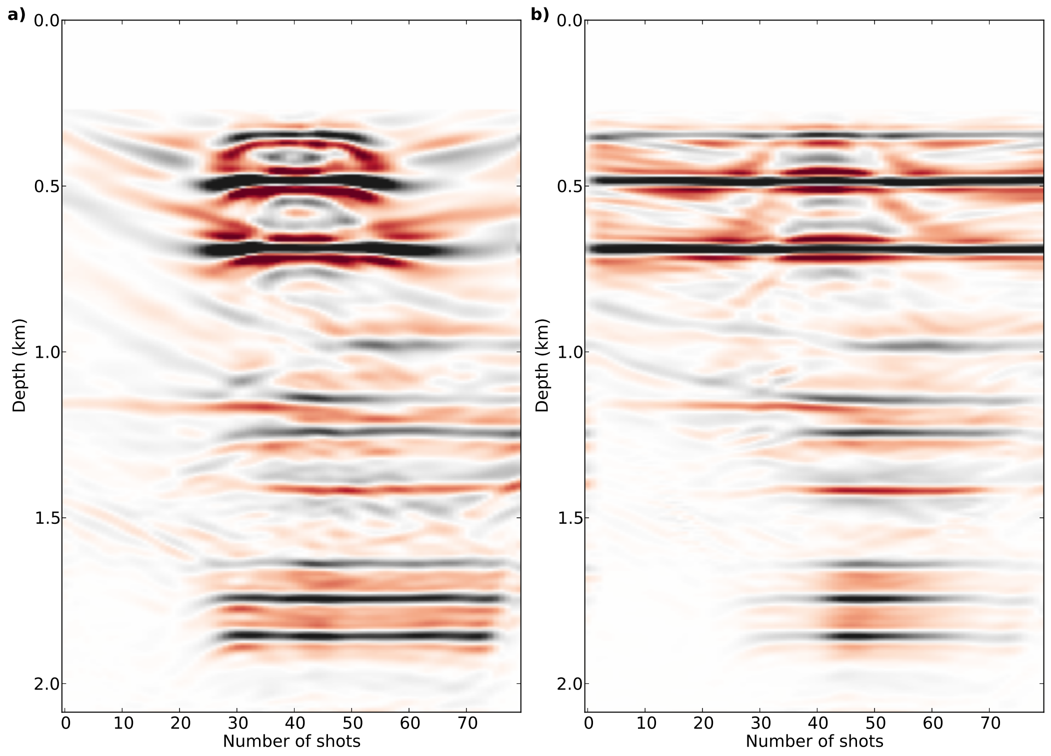

Figure 3.13: Shot-index least-squares migration image at $x = 3.75$ km. (a) Shot-index image of $\delta\lambda$. Shot-index image of $\delta\mu$.

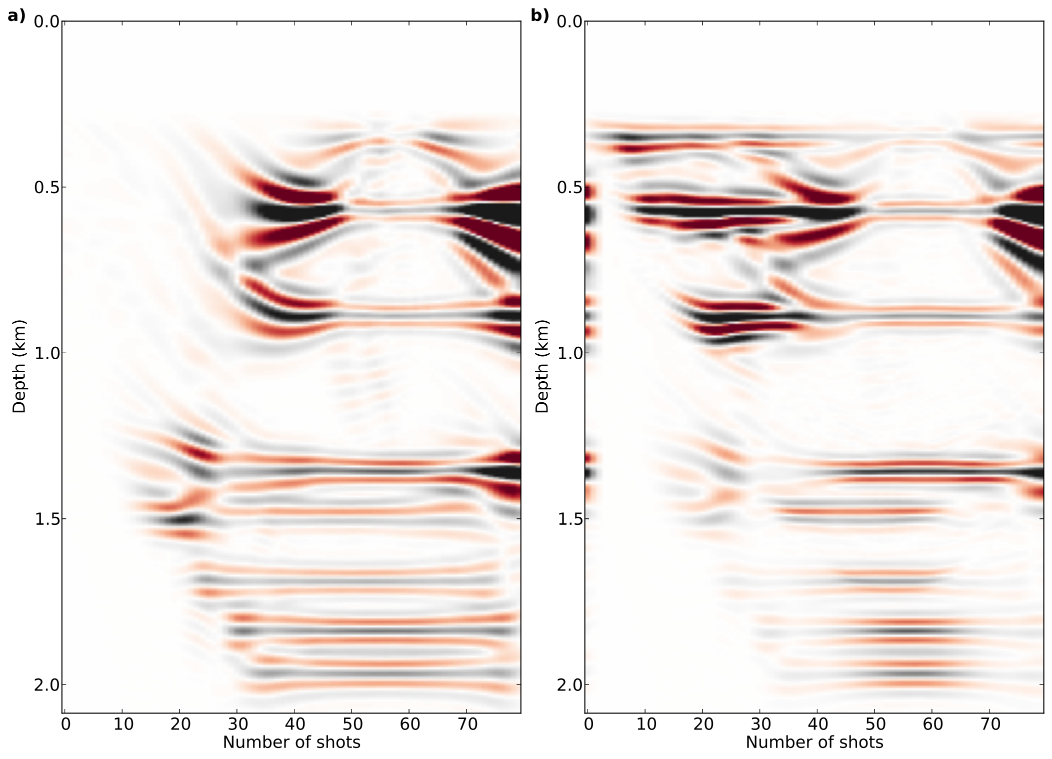

Figure 3.14: Shot-index migration image at $x = 5.25$ km. (a) Shot-index image of $\delta\lambda$. (b) Shot-index image of $\delta\mu$.

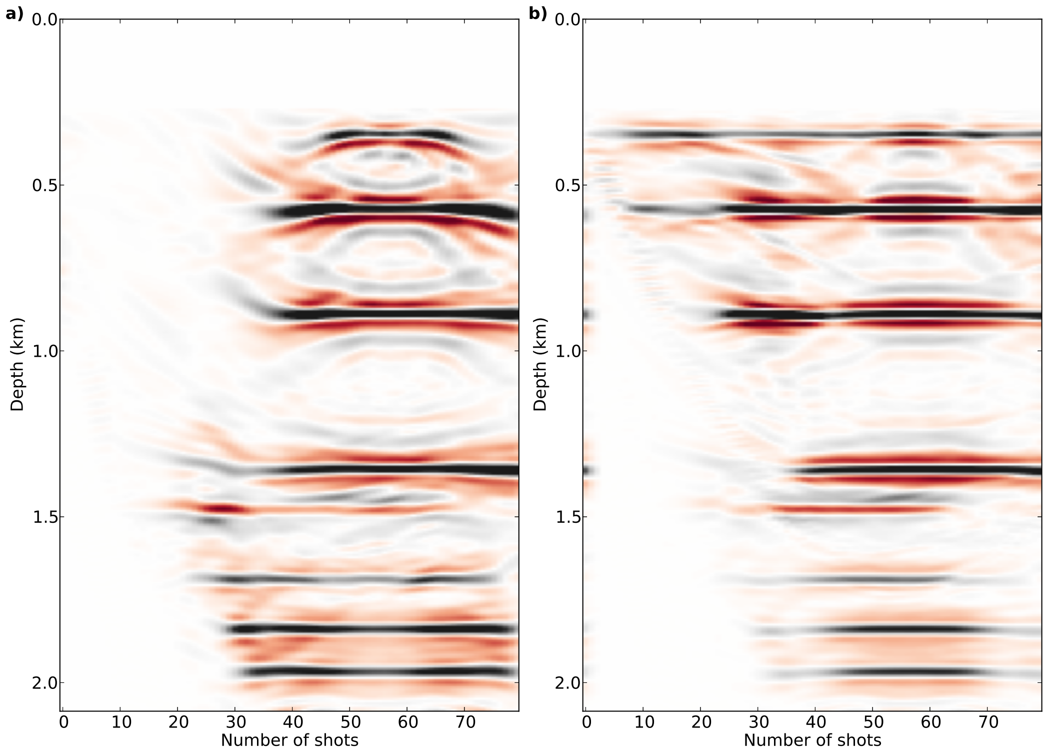

Figure 3.15: Shot-index least-squares migration image at $x = 5.25$ km. (a) Shot-index image of $\delta\lambda$. (b) Shot-index image of $\delta\mu$.

CHAPTER 4

4.1 Main conclusions

Conventionally, the problem of least-squares migration has been mainly investigated via one-wave operators and ray-based techniques. In this thesis, least-squares migration is studied in terms of the acoustic and elastic Born approximation and Green's functions of the two-way wave equation. A clear workflow is presented to construct operators in the acoustic and elastic two-way wave equation migration. The problem is solved by the conjugate gradient method, which requires access to two fundamental computer codes: the forward and its exact adjoint operators. To my knowledge, this thesis presents, for the first time, a step-by-step procedure to design forward operator and its exact adjoint operator for both acoustic and elastic two-way wave equation least-squares migration in the time domain. Many research groups have reported results of the least-squares two-way wave equation migration with an adjoint of the classic reverse time migration operator. However, the issue of adjointness is never fully disclosed in the geophysical literature dealing with the least-squares migration of two-way wave equation. This research provides a clear path to develop computer codes for the least-squares two-way wave equation migration that are capable of passing the dot-product test in machine precision.

I conclude that a classical reverse time migration code is not the exact adjoint of the forward operator [1]. If the inexact adjoint operator is adopted, the conjugate gradient method leads to an algorithm that does not converge. In this situation, one could adopt the method of steepest descent or non-linear conjugate gradient (Yao, 2013) to estimate the least-squares migrated image. However, steepest descent and non-linear conjugate gradients require a

[1] It is the main reason that to denominate the problem as least-squares two-way wave equation migration instead of least-squares reverse time migration.

line search to estimate step-lengths in each iteration. The beauty of the conjugate gradients method is that step-lengths are analytically computed (there is no need of a line search). The latter is the main reason I advise to design the adjoint operator that does pass the dot-product test.

Adopting a process that entails computing operators based on two-way wave equation avoids some of the problems of least-squares migration with one-way wave equation operators. For instance, the proposed algorithm should have an optimal response to steep structures and the ability to image turning waves. In general, our algorithm combines the benefits of reverse time migration in terms of being able to handle data arising from complex structures. It is obvious that one shortcoming of the least-squares migration is cost. To accelerate the convergence rate of least squares migration, preconditioning operators is incorporated in the formulation of the least-squares problem to improve the distribution of amplitudes in the shallow and deep areas. Moreover, a regularization term is also added to the least-squares problem, which permits us to add constraints to the least-squares problem to reduce artifact in the multi-shot examples for both acoustic and elastic seismic imaging.

Overall, the least-squares two-way wave equation migration is an advanced migration method to generate a high resolution and artifact-free images. The field of linearized inversion-based migration offers an interesting playground to test regularization and preconditioning methods for improving the quality of seismic images. It also provides an interesting area for testing forward and adjoint operators that can become part of full-waveform inversion methodologies.

4.2 Main contribution

The main contribution of this thesis includes elastic response kernel, the exact adjoint operator and the framework of least-squares migration in the matrix formulation. The response kernel is derived from linearized two-way wave equation of elastic media. The kernels define a clear relation between model perturbation and the associated subsurface responses. The adjoint of the elastic kernel can successfully avoid polarity issue and therefore, permits us to stack multiple shot-profile images without the expensive process of polarity correction.

This research also investigates the exact adjoint adjointness of the discretized forward and adjoint operators for both acoustic and elastic least-squares two-way wave equation migration. The proposed exact adjoint operator can precisely pass the dot-product test. In this research, it is stressed that, in the discrete domain, the exact adjoint operator is not same as the classic reverse time migration and the reverse time migration is the pseudo-adjoint of the linearized wave equation. Besides, for the sake of memory efficiency and computational speed, the forward operator and its adjoint operator are implemented in the form of a matrix-free algorithm.

Eventually, the correct adjointness permits us to adopt conjugate gradient method to solve the problem of least-squares migration. The inversion-based migration provides an access to estimate the linearized relation between data perturbation and model perturbation. The linearized least-squares migration can be utilized to construct a bridge to investigate the more complicated non-linear data-model inversion (e.g. full waveform inversion) in the future.

4.3 Future developments

Computational cost is one of the concerns in the least-squares two-way wave-equation migration. The iterative algorithm is expensive in terms of computational time and hardware requirements (memory). Its implementation in 3D and/or on large 2D data sets requires serious consideration given our group current computational infrastructure. Several research directions should be followed to further advance the field of least-squares migration. For instance, future work could move the field in the following direction:

development efficient 3D acoustic and elastic codes to process real data; and,

application of least-squares two-way wave equation migration to global seismology.

The development of 3D codes is limited by access to hardware. At the present time, the algorithm infrastructure develops in this thesis is ready to be moved to cope with 3D problems. However, several aspects pertaining preconditioning of 3D survey might have to be considered. For instance, regularization of receivers, denoising, source amplitude equalization, etc. Preconditioning problems might be extremely severe for onshore data. For marine data, access to high-quality demultiple data might also be important as our methods require data that are not contaminated by multiples.

Moving the problem to global seismology for regional studies might also involve some interesting challenges. Data are sparse and seismic sources are highly variable. Equalization of sources, denoising and proper selection of data might also be important to assemble a realistic volume that can be used to test teleseismic least-squares migration.

Bibliography

Baysal, E., Kosloff, D. D., and Sherwood, J. W. C. (1983). Reverse time migration. *Geophysics*, 48(11):1514–1524.

Berenger, J.-P. (1996). Perfectly matched layer for the FDTD solution of wave-structure interaction problems. *IEEE Transactions on Antennas and Propagation*, 44(1):110–117.

Červenỳ, V., Popov, M. M., and Pšenčík, I. (1982). Computation of wave fields in inhomogeneous media – Gaussian beam approach. *Geophysical Journal International*, 70(1):109–128.

Claerbout, J. F. (1971). Toward a unified theory of reflector mapping. *Geophysics*, 36(3):467–481.

Claerbout, J. F. (1992). Earth sounding analysis: Processing versus inversion. *Blackwell Scientific Publication*.

Clapp, M. L. (2005). *Imaging under salt: Illumination compensation by regularized inversion*. PhD thesis, Stanford University.

da Costa, C. A., Raz, S., and Kosloff, D. (1989). Gaussian beam migration. *SEG Technical Program Expanded Abstracts*, pages 1169–1171.

Dablain, M. A. (1986). The application of high-order differencing to the scalar wave equation. *Geophysics*, 51(1):54–66.

Du, Q., Zhu, Y., and Ba, J. (2012). Polarity reversal correction for elastic reverse time migration. *Geophysics*, 77(2):S31–S41.

Duan, Y., Sava, P., and Guitton, A. (2016). Elastic least-squares reverse time migration. *SEG Technical Program Expanded Abstracts*, pages 4152–4157.

Feng, Z. and Schuster, G. (2016). Elastic least-squares reverse time migration. *SEG Technical Program Expanded Abstracts*, pages 4163–4167.

Gardner, G. H. F., Gardner, L. W., and Gregory, A. R. (1974). Formation velocity and density—the diagnostic basics for stratigraphic traps. *Geophysics*, 39(6):770–780.

Gazdag, J. (1978). Wave equation migration with the phase-shift method. *Geophysics*, 43(7):1342–1351.

Gazdag, J. and Sguazzero, P. (1984). Migration of seismic data by phase shift plus interpolation. *Geophysics*, 49(2):124–131.

Gray, S. H. (1986). Efficient traveltime calculations for Kirchhoff migration. *Geophysics*, 51(8):1685–1688.

Guitton, A., Kaelin, B., and Biondi, B. (2007). Least-squares attenuation of reverse-time-migration artifacts. *Geophysics*, 72(1):S19–S23.

Guitton, A., Valenciano, A., and Bevc, D. (2006). Robust imaging condition for shot-profile migration. *SEG Technical Program Expanded Abstracts*, pages 2519–2523.

Hale, D. (1992). *Migration by the Kirchhoff, slant stack, and Gaussian beam methods.* Colorado School of Mines, Golden, CO (United States). Center for Wave Phenomena.

Hansen, P. C. (1987). *Rank-deficient and discrete ill-posed problems: numerical aspects of linear inversion.* Society for Industrial Mathematics.

Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49.

Hill, N. R. (1990). Gaussian beam migration. *Geophysics*, 55(11):1416–1428.

Huang, Y., Nammour, R., and Symes, W. (2016). Flexibly preconditioned extended least-squares migration in shot-record domain. *Geophysics*, 81(5):S299–S315.

Ji, J. (2009). An exact adjoint operation pair in time extrapolation and its application in least-squares reverse-time migration. *Geophysics*, 74(5):H27–H33.

Kessinger, W. (1992). Extended split-step Fourier migration. *SEG Technical Program Expanded Abstracts*, pages 917–920.

Keys, R. and Weglein, A. (1983). Generalized linear inversion and the first Born theory for acoustic media. *Journal of Mathematical Physics*, 24(6):1444–1449.

Kühl, H. and Sacchi, M. D. (2003). Least-squares wave-equation migration for AVP/AVA inversion. *Geophysics*, 68(1):262–273.

Lailly, P. (1983). The seismic inverse problem as a sequence of before stack migrations. *Society of Industrial and Applied Mathematics, Expanded Abstracts*, pages 206–220.

Lazaratos, S. K. and Harris, J. M. (1990). Radon transform/Gaussian beam migration. *SEG Technical Program Expanded Abstracts*, pages 1314–1317.

Levander, A. R. (1988). Fourth-order finite-difference P-SV seismograms. *Geophysics*, 53(11):1425–1436.

Levin, S. A. (1984). Principle of reverse-time migration. *Geophysics*, 49(5):581–583.

Loewenthal, D. and Mufti, I. R. (1983). Reversed time migration in spatial frequency domain. *Geophysics*, 48(5):627–635.

Lu, R., Yan, J., Traynin, P., Anderson, J. E., and Dickens, T. (2010). Elastic RTM: anisotropic wave-mode separation and converted-wave polarization correction. *SEG Technical Program Expanded Abstracts*, pages 3171–3175.

Malvern, L. (1969). *Introduction to the mechanics of a continuous medium.* Prentice-Hall series in engineering of the physical sciences. Prentice-Hall.

McMechan, G. (1983). Migration by extrapolation of time-dependent boundary values. *Geophysical Prospecting*, 31(3):413–420.

Mora, P. (1987). Nonlinear two-dimensional elastic inversion of multioffset seismic data. *Geophysics*, 52(9):1211–1228.

Mora, P. (1988). Elastic wave-field inversion of reflection and transmission data. *Geophysics*, 53(6):750–759.

Nemeth, T., Wu, C., and Schuster, G. T. (1999). Least-squares migration of incomplete reflection data. *Geophysics*, 64(1):208–221.

Popovici, A. M. (1996). Prestack migration by split-step DSR. *Geophysics*, 61(5):1412–1416.

Ronen, S., Nichols, D., Bale, R., and Ferber, R. (1995). Dealiasing dmo: Good-pass, bad-pass, and unconstrained. *SEG Technical Program Expanded Abstracts 1995*, pages 743–746.

Rosales, D. and Rickett, J. (2001). PS-wave polarity reversal in angle domain common-image gathers. *SEG Technical Program Expanded Abstracts*, pages 1843–1846.

Rosales, D. A., Fomel, S., Biondi, B. L., and Sava, P. C. (2008). Wave-equation angle-domain common-image gathers for converted waves. *Geophysics*, 73(1):S17–S26.

Schneider, J. B., Wagner, C. L., and Broschat, S. L. (1998). Implementation of transparent sources embedded in acoustic finite-difference time-domain grids. *The Journal of the Acoustical Society of America*, 103(1):136–142.

Stoffa, P. L., Fokkema, J. T., de Luna Freire, R. M., and Kessinger, W. P. (1990). Split-step Fourier migration. *Geophysics*, 55(4):410–421.

Sun, Y., Qin, F., Checkles, S., and Leveille, J. P. (2000). 3-D prestack Kirchhoff beam migration for depth imaging. *Geophysics*, 65(5):1592–1603.

Symes, W. W. (2008). Approximate linearized inversion by optimal scaling of prestack depth migration. *Geophysics*, 73(2):R23–R35.

Tarantola, A. (1984). Inversion of seismic reflection data in the acoustic approximation. *Geophysics*, 49(8):1259–1266.

Valenciano, A. A. and Biondi, B. (2003). 2-D deconvolution imaging condition for shot-profile migration. *SEG Technical Program Expanded Abstracts*, pages 1059–1062.

Valenciano, A. A., Biondi, B., and Guitton, A. (2006). Target-oriented wave-equation inversion. *Geophysics*, 71(4):A35–A38.

Virieux, J. (1984). SH-wave propagation in heterogeneous media: Velocity-stress finite-difference method. *Geophysics*, 49(11):1933–1942.

Wang, J., Kuehl, H., and Sacchi, M. D. (2005). High-resolution wave-equation AVA imaging: Algorithm and tests with a data set from the Western Canadian Sedimentary Basin. *Geophysics*, 70(5):S91–S99.

Whitmore, N. D. (1983). Iterative depth migration by backward time propagation. *SEG Technical Program Expanded Abstracts*, pages 382–385.

Yan, J. and Sava, P. (2008). Isotropic angle-domain elastic reverse-time migration. *Geophysics*, 73(6):S229–S239.

Yang, J., Liu, Y., and Dong, L. (2016). Least-squares reverse time migration in the presence of density variations. *Geophysics*, 81(6):S497–S509.

Yao, G. (2013). *Least-squares reverse-time migration.* PhD Thesis, Imperial College London, UK.

Yao, G. and Jakubowicz, H. (2015). Least-squares reverse-time migration in a matrix-based formulation. *Geophysical Prospecting*, 64(3):611–621.

Zeng, C., Dong, S., Mao, J., and Wang, B. (2014). Broadband least-squares reverse time migration for complex structure imaging. *SEG Technical Program Expanded Abstracts*, pages 3715–3719.

Zhang, Y., Duan, L., and Xie, Y. (2015). A stable and practical implementation of least-squares reverse time migration. *Geophysics*, 80(1):V23–V31.

APPENDIX A

Discretization of Partial Differential Equations

A.1 Numerical Simulation of Acoustic Wave Equation with PML Boundary Condition

Acoustic wave equation consists of three first-order ordinary differential equations (Malvern, 1969) as

$$\begin{cases} \frac{\partial v_x}{\partial t} = \frac{1}{\rho}\frac{\partial p}{\partial x}, \\ \frac{\partial v_z}{\partial t} = \frac{1}{\rho}\frac{\partial p}{\partial z}, \\ \frac{\partial p}{\partial t} = \lambda(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}). \end{cases} \tag{A.1}$$

Due to the theory of perfectly matched layer (Berenger, 1996), we virtually split the pressure component into horizontal and vertical directions as $p = p_x + p_z$ (Yao, 2013) and substitute it into equation A.1 as

$$\begin{cases} \frac{\partial v_x}{\partial t} = \frac{1}{\rho}\frac{\partial p}{\partial x}, \\ \frac{\partial v_z}{\partial t} = \frac{1}{\rho}\frac{\partial p}{\partial z}, \\ \frac{\partial p_x}{\partial t} = \lambda\frac{\partial v_x}{\partial x}, \\ \frac{\partial p_z}{\partial t} = \lambda\frac{\partial v_z}{\partial z}, \\ p = p_x + p_z. \end{cases} \tag{A.2}$$

Equation A.2 contains four ordinary differential equations, which permits us to incorporate perfect matched layer as boundary condition (Berenger, 1996) to attenuate artificial reflections from numerical boundaries. Then we have

$$\begin{cases} \frac{\partial v_x}{\partial t} + \kappa_x v_x = \frac{1}{\rho}\frac{\partial p}{\partial x}, \\ \frac{\partial v_z}{\partial t} + \kappa_z v_z = \frac{1}{\rho}\frac{\partial p}{\partial z}, \\ \frac{\partial p_x}{\partial t} + \kappa_x p_x = \lambda\frac{\partial v_x}{\partial x}, \\ \frac{\partial p_z}{\partial t} + \kappa_z p_z = \lambda\frac{\partial v_z}{\partial z}, \\ p = p_x + p_z ., \end{cases} \tag{A.3}$$

where κ_x and κ_z are attenuation coefficients to attenuate artificial reflections from vertical and horizontal boundaries respectively (Berenger, 1996). Equation A.3 can be solved numerically by finite difference method, which includes two steps: variable quantization and differential operator discretization. The variables are quantized on a mesh grid. As equation A.3 contains multiple variables, it is important to properly distribute each component onto the mesh-grid. We uniformly quantize and distribute pressure wavefield on a 3D cubic mesh-grid of size $nx \times nz \times nt$. On the mesh-grid, any pressure point can be presented as $p(ix, iz, it)$, where $ix \in \{1, 2, 3, ..., nx\}$, $iz \in \{1, 2, 3, ..., nz\}$ and $it \in \{1, 2, 3, ..., nt\}$. p_x and p_z must share the same position with p since they are the virtual subsets of pressure. To obtain vertical velocity in the first ordinary differential equation of A.3, we adopt central finite difference method (Dablain, 1986). Based on the definition of central difference, each point in the vertical velocity wavefield is located in the middle of every two adjacent points in pressure wavefield along horizontal (x) and temporal (t) axis. In other words, each vertical velocity point should be presented as $v_x(ix + 0.5, iz, it + 0.5)$. Similarly, the second ordinary differential equation of A.3 indicates that the vertical velocity should be presented as $v_z(ix, iz + 0.5, it + 0.5)$. In this research, we use second-order and higher-order central difference to discretize temporal and spatial differential operators (Dablain, 1986). Therefore, equation A.3 can be discretized as

$$\begin{cases} v_x(ix+0.5, iz, it+0.5) = A_1 v_x(ix+0.5, iz, it-0.5) + \cdots \\ \qquad +B_1\{\sum_1^N c_n[-p(ix-(n-1), iz, it) + p(ix+n, iz, it)]\}, \\ v_z(ix, iz+0.5, it+0.5) = A_2 v_z(ix, iz+0.5, it-0.5) + \cdots \\ \qquad +B_2\{\sum_1^N c_n[-p(ix, iz-(n-1), it) + p(ix, iz+n, it)]\}, \\ p_x(ix, iz, it) = A_3 p_x(ix, iz, it-1) + \cdots \\ \qquad +B_3\{\sum_1^N c_n[v_x(ix-0.5+n, iz, it+0.5) - v_x(ix+0.5+n, iz, it+0.5)]\}, \\ p_z(ix, iz, it) = A_4 p_z(ix, iz, it-1) + \cdots \\ \qquad +B_4\{\sum_1^N c_n[v_z(ix, iz-0.5+n, it+0.5) - v_z(ix, iz+0.5+n, it+0.5)]\}, \\ p(ix, iz, it) = p_x(ix, iz, it) + p_z(ix, iz, it), \end{cases} \tag{A.4}$$

where

$$\begin{cases} A_1(ix+0.5,iz) = \frac{2-\Delta t\kappa_x(ix+0.5,iz)}{2+\Delta t\kappa_x(ix+0.5,iz)}, \\ A_2(ix,iz+0.5) = \frac{2-\Delta t\kappa_z(ix,iz+0.5)}{2+\Delta t\kappa_z(ix,iz+0.5)}, \\ A_3(ix,iz) = \frac{2-\Delta t\kappa_x(ix,iz)}{2+\Delta t\kappa_x(ix,iz)}, \\ A_4(ix,iz) = \frac{2-\Delta t\kappa_z(ix,iz)}{2+\Delta t\kappa_z(ix,iz)}, \\ B_1(ix+0.5,iz) = \frac{2\Delta t}{\rho(ix+0.5,iz)\Delta x[2+\Delta t\kappa_x(ix+0.5,iz)]}, \\ B_2(ix,iz+0.5) = \frac{2\Delta t}{\rho(ix,iz+0.5)\Delta x[2+\Delta t\kappa_z(ix,iz+0.5)]}, \\ B_3(ix,iz) = \frac{2\Delta t\lambda(ix,iz)}{\Delta x[2+\Delta t\kappa_x(ix,iz)]}, \\ B_4(ix,iz) = \frac{2\Delta t\lambda(ix,iz)}{\Delta x[2+\Delta t\kappa_z(ix,iz)]}. \end{cases} \tag{A.5}$$

The equation A.5 are the diagonal operators in equation 2.15. Finally, we can summarize the discretization of acoustic wave equation into four steps:

- virtually split pressure components and add attenuation terms to each ordinary equation based on the theory of perfectly matched layer;
- discretize pressure on a mesh-grid in space and time and mark it as (ix, iz, it);
- find the location of other components with respect to the location of pressure component based on central difference;
- use finite difference method to discretize temporal and spatial derivative operators, respectively.

A.2 Numerical Simulation of Elastic Wave Equation with PML Boundary Condition

Elastic wave equation consists of 5 partial differential equations (Malvern, 1969) as

$$\begin{cases} \frac{\partial v_x}{\partial t} = \frac{1}{\rho}(\frac{\partial \tau_x}{\partial x} + \frac{\partial \tau_s}{\partial z}), \\ \frac{\partial v_z}{\partial t} = \frac{1}{\rho}(\frac{\partial \tau_z}{\partial z} + \frac{\partial \tau_s}{\partial x}), \\ \frac{\partial \tau_x}{\partial t} = \lambda(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}) - 2\mu\frac{\partial v_x}{\partial x}, \\ \frac{\partial \tau_z}{\partial t} = \lambda(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}) - 2\mu\frac{\partial v_z}{\partial z}, \\ \frac{\partial \tau_s}{\partial t} = \mu(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x}). \end{cases} \tag{A.6}$$

To attenuate artificial boundary reflections, we adopt perfect matched layer (Berenger, 1996) as boundary condition in equation A.6. To incorporate attenuation coefficients of the boundary condition, we convert the partial differential equations of A.6 into ordinary differential equations via virtually spliting stress and strain components as

$$\begin{cases} \frac{\partial v_x^x}{\partial t} = \frac{1}{\rho_0}\frac{\partial \tau_x}{\partial x}, \\ \frac{\partial v_x^z}{\partial t} = \frac{1}{\rho_0}\frac{\partial \tau_s}{\partial z}, \\ v_x = v_x^x + v_x^z, \end{cases} \tag{A.7}$$

$$\begin{cases} \frac{\partial v_z^x}{\partial t} = \frac{1}{\rho_0}\frac{\partial \delta\tau_s}{\partial x}, \\ \frac{\partial v_z^z}{\partial t} = \frac{1}{\rho_0}\frac{\partial \delta\tau_z}{\partial z}, \\ v_z = v_z^x + v_z^z, \end{cases} \tag{A.8}$$

$$\begin{cases} \frac{\partial \tau_x^x}{\partial t} = (\lambda_0 + 2\mu_0)\frac{\partial v_x}{\partial x}, \\ \frac{\partial \tau_x^z}{\partial t} = \lambda_0\frac{\partial v_z}{\partial z}, \\ \tau_x = \tau_x^x + \tau_x^z, \end{cases} \tag{A.9}$$

$$\begin{cases} \frac{\partial \tau_z^x}{\partial t} = \lambda_0\frac{\partial v_x}{\partial x}, \\ \frac{\partial \tau_z^z}{\partial t} = (\lambda_0 + 2\mu_0)\frac{\partial v_z}{\partial z}, \\ \tau_z = \tau_z^x + \tau_z^z, \end{cases} \tag{A.10}$$

$$\begin{cases} \frac{\partial \tau_s^x}{\partial t} = \mu_0\frac{\partial v_z}{\partial x}, \\ \frac{\partial \tau_s^z}{\partial t} = \mu_0\frac{\partial v_x}{\partial z}, \\ \tau_s = \tau_s^x + \tau_s^z. \end{cases} \tag{A.11}$$

Based on the method of perfectly matched layer (Berenger, 1996), attenuation coefficients are incorporated into the ordinary differential equations above and we obtain

$$\begin{cases} \frac{\partial v_x^x}{\partial t} + \kappa_x v_x^x = \frac{1}{\rho_0}\frac{\partial \tau_x}{\partial x}, \\ \frac{\partial v_x^z}{\partial t} + \kappa_z v_x^z = \frac{1}{\rho_0}\frac{\partial \tau_s}{\partial z}, \\ v_x = v_x^x + v_x^z, \end{cases} \tag{A.12}$$

$$\begin{cases} \frac{\partial v_z^x}{\partial t} + \kappa_x v_z^x = \frac{1}{\rho_0}\frac{\partial \delta\tau_s}{\partial x}, \\ \frac{\partial v_z^z}{\partial t} + \kappa_z v_z^z = \frac{1}{\rho_0}\frac{\partial \delta\tau_z}{\partial z}, \\ v_z = v_z^x + v_z^z, \end{cases} \tag{A.13}$$

$$\begin{cases} \frac{\partial \tau_x^x}{\partial t} + \kappa_x \tau_x^x = (\lambda_0 + 2\mu_0)\frac{\partial v_x}{\partial x}, \\ \frac{\partial \tau_x^z}{\partial t} + \kappa_z \tau_x^z = \lambda_0\frac{\partial v_z}{\partial z}, \\ \tau_x = \tau_x^x + \tau_x^z. \end{cases} \tag{A.14}$$

$$\begin{cases} \frac{\partial \tau_z^x}{\partial t} + \kappa_x \tau_z^x = \lambda_0\frac{\partial v_x}{\partial x}, \\ \frac{\partial \tau_z^z}{\partial t} + \kappa_z \tau_z^z = (\lambda_0 + 2\mu_0)\frac{\partial v_z}{\partial z}, \\ \tau_z = \tau_z^x + \tau_z^z. \end{cases} \tag{A.15}$$

$$\begin{cases} \frac{\partial \tau_s^x}{\partial t} + \kappa_x \tau_s^x = \mu_0\frac{\partial v_z}{\partial x}, \\ \frac{\partial \tau_s^z}{\partial t} + \kappa_z \tau_s^z = \mu_0\frac{\partial v_x}{\partial z}, \\ \tau_s = \tau_s^x + \tau_s^z, \end{cases} \tag{A.16}$$

where κ_x and κ_z are attenuation coefficients to attenuate artificial reflections from vertical and horizontal boundaries respectively (Berenger, 1996). Elastic wave equation (A.12 to A.16) can be solved numerically by finite difference method, which includes two steps: variable quantization and differential operator discretization. The variables are quantized on a mesh grid. To solve the elastic wave equation in computer, it is importance to properly allocate each component on the mesh grid. First, we uniformly discretize and allocate horizontal stress τ_x on a 3D mesh grid of size $nx \times nz \times nt$. The grid permit us to present the horizontal stress field and its virtual components as $\tau_x(ix, iz, it)$, $\tau_x^x(ix, iz, it)$ and $\tau_x^z(ix, iz, it)$, where $ix \in \{1, 2, 3, ..., nx\}$, $iz \in \{1, 2, 3, ..., nz\}$ and $it \in \{1, 2, 3, ..., nt\}$. We adopts central finite difference method (Levander, 1988; Virieux, 1984) to solve the horizontal stress field τ_x and its virtual components τ_x^x and τ_x^z in equation A.14. Based on the definition of central difference, each point in the virtual stress field τ_x^x should be in the middle of the two adjacent points in v_x along horizontal axis and time axis, so the horizontal velocity should be expressed as $v_x(ix + 0.5, iz, it + 0.5)$. Similarly, vertical velocity should be located at $v_z(ix, iz+0.5, it+0.5)$. Based on $v_x(ix+0.5, iz, it+0.5)$ and $v_z(ix, iz+0.5, it+0.5)$, equation A.16 indicates the location of shear stress, which can be presented as $\tau_s(ix+0.5, iz+0.5, it)$. According to equation A.14 and A.15, the vertical stress should share the same location with the horizontal stress on the mesh grid because both horizontal and vertical stresses are obtained by calculating $\frac{\partial v_x}{\partial x}$ and $\frac{\partial v_z}{\partial z}$. Therefore, the vertical stress should be presented as $\tau_z(ix, iz, it)$. In this research, we use second-order and higher-order central difference to discretize temporal and spatial differential operators. Therefore, the entire elastic wave equation can be discretized as

$$\begin{cases} v_x^x(ix + 0.5, iz, it + 0.5) = A_1^{v_x} v_x^x(ix + 0.5, iz, it - 0.5) + \cdots \\ \qquad + B_1^{v_x}\{\sum_1^N c_n[-\tau_x(ix - (n-1), iz, it + 0.5) + \tau_x(ix + n, iz, it)]\}, \\ v_x^z(ix + 0.5, iz, it + 0.5) = A_2^{v_x} v_x^z(ix + 0.5, iz, it - 0.5) + \cdots \\ \qquad + B_2^{v_x}\{\sum_1^N c_n[-\tau_s(ix, iz - (n-1), it + 0.5) + \tau_s(ix, iz + n, it)]\}, \\ v_x(ix + 0.5, iz, it + 0.5) = v_x^x(ix + 0.5, iz, it + 0.5) + v_x^z(ix + 0.5, iz, it + 0.5), \end{cases} \tag{A.17}$$

$$\begin{cases} v_z^x(ix, iz + 0.5, it + 0.5) = A_1^{v_x} v_z^x(ix, iz + 0.5, it - 0.5) + \cdots \\ \qquad + B_1^{v_z}\{\sum_1^N c_n[-\tau_s(ix - (n-1), iz, it + 0.5) + \tau_s(ix + n, iz, it)]\}, \\ v_z^z(ix, iz + 0.5, it + 0.5) = A_2^{v_x} v_z^z(ix, iz + 0.5, it - 0.5) + \cdots \\ \qquad + B_2^{v_z}\{\sum_1^N c_n[-\tau_z(ix, iz - (n-1), it + 0.5) + \tau_z(ix, iz + n, it)]\}, \\ v_z(ix, iz + 0.5, it + 0.5) = v_z^x(ix, iz + 0.5, it + 0.5) + v_z^z(ix, iz + 0.5, it + 0.5), \end{cases} \tag{A.18}$$

$$\begin{cases} \tau_x^x(ix,iz,it) = A_1^{\tau_x}\tau_x^x(ix,iz,it-1) + \cdots \\ \qquad +B_1^{\tau_x}\{\sum_1^N c_n[v_x(ix-0.5+(n-1),iz,it-0.5) - v_x(ix+0.5+n,iz,it-0.5)]\}, \\ \tau_x^z(ix,iz,it) = A_2^{\tau_x}\tau_x^z(ix,iz,it-1) + \cdots \\ \qquad +B_2^{\tau_x}\{\sum_1^N c_n[v_z(ix,iz-0.5+(n-1),it-0.5) - v_z(ix,iz+0.5+n,it-0.5)]\}, \\ \tau_x(ix,iz,it) = \tau_x^x(ix,iz,it) + \tau_x^z(ix,iz,it), \end{cases} \tag{A.19}$$

$$\begin{cases} \tau_z^x(ix,iz,it) = A_1^{\tau_z}\tau_z^x(ix,iz,it-1) + \cdots \\ \qquad +B_1^{\tau_z}\{\sum_1^N c_n[v_x(ix-0.5+(n-1),iz,it-0.5) - v_x(ix+0.5+n,iz,it-0.5)]\}, \\ \tau_z^z(ix,iz,it) = A_2^{\tau_z}\tau_z^z(ix,iz,it-1) + \cdots \\ \qquad +B_2^{\tau_z}\{\sum_1^N c_n[v_z(ix,iz-0.5+(n-1),it-0.5) - v_z(ix,iz+0.5+n,it-0.5)]\}, \\ \tau_z(ix,iz,it) = \tau_z^x(ix,iz,it) + \tau_z^z(ix,iz,it), \end{cases} \tag{A.20}$$

$$\begin{cases} \tau_s^x(ix+0.5,iz+0.5,it) = A_1^{\tau_s}\tau_s^x(ix+0.5,iz+0.5,it-1) + \cdots \\ \qquad +B_1^{\tau_s}\{\sum_1^N c_n[v_x(ix+0.5,iz-(n-1),it-0.5) - v_x(ix+0.5,iz+n,it-0.5)]\}, \\ \tau_s^z(ix+0.5,iz+0.5,it) = A_2^{\tau_s}\tau_s^z(ix+0.5,iz+0.5,it-1) + \cdots \\ \qquad +B_2^{\tau_s}\{\sum_1^N c_n[v_z(ix-(n-1),iz+0.5,it-0.5) - v_z(ix+n,iz+0.5,it-0.5)]\}, \\ \tau_s(ix+0.5,iz+0.5,it) = \tau_s^x(ix+0.5,iz+0.5,it) + \tau_s^z(ix+0.5,iz+0.5,it), \end{cases} \tag{A.21}$$

where

$$\begin{cases}
A_1^{v_x}(ix+0.5,iz) = \frac{2-\Delta t\kappa_x(ix+0.5,iz)}{2+\Delta t\kappa_x(ix+0.5,iz)} \\
A_2^{v_x}(ix+0.5,iz) = \frac{2-\Delta t\kappa_z(ix+0.5,iz)}{2+\Delta t\kappa_z(ix+0.5,iz)} \\
A_1^{v_z}(ix,iz+0.5) = \frac{2-\Delta t\kappa_x(ix,iz+0.5)}{2+\Delta t\kappa_x(ix+0.5,iz)} \\
A_2^{v_z}(ix,iz+0.5) = \frac{2-\Delta t\kappa_z(ix,iz+0.5)}{2+\Delta t\kappa_z(ix,iz+0.5)} \\
A_1^{\tau_x}(ix,iz) = \frac{2-\Delta t\kappa_x(ix,iz)}{2+\Delta t\kappa_x(ix,iz)} \\
A_2^{\tau_x}(ix,iz) = \frac{2-\Delta t\kappa_z(ix,iz)}{2+\Delta t\kappa_x(ix,iz)} \\
A_1^{\tau_z}(ix,iz) = \frac{2-\Delta t\kappa_x(ix,iz)}{2+\Delta t\kappa_x(ix,iz)} \\
A_2^{\tau_z}(ix,iz) = \frac{2-\Delta t\kappa_z(ix,iz)}{2+\Delta t\kappa_x(ix,iz)} \\
A_1^{\tau_s}(ix+0.5,iz+0.5) = \frac{2-\Delta t\kappa_x(ix+0.5,iz+0.5)}{2+\Delta t\kappa_x(ix+0.5,iz+0.5)} \\
A_2^{\tau_s}(ix+0.5,iz+0.5) = \frac{2-\Delta t\kappa_z(ix+0.5,iz+0.5)}{2+\Delta t\kappa_x(ix+0.5,iz+0.5)} \\
B_1^{v_x}(ix+0.5,iz) = \frac{2\Delta t}{\rho(ix+0.5,iz)\Delta x[2+\Delta t\kappa_x(ix+0.5,iz)]} \\
B_2^{v_x}(ix+0.5,iz) = \frac{2\Delta t}{\rho(ix+0.5,iz)\Delta z[2+\Delta t\kappa_z(ix+0.5,iz)]} \\
B_1^{v_z}(ix,iz+0.5) = \frac{2\Delta t}{\rho(ix,iz+0.5)\Delta x[2+\Delta t\kappa_x(ix,iz+0.5)]} \\
B_2^{v_z}(ix,iz+0.5) = \frac{2\Delta t}{\rho(ix,iz+0.5)\Delta z[2+\Delta t\kappa_z(ix,iz+0.5)]} \\
B_1^{\tau_x}(ix,iz) = \frac{2\Delta t[\lambda(ix,iz)+2\mu(ix,iz)]}{\Delta x[2+\Delta t\kappa_z(ix,iz)]} \\
B_2^{\tau_x}(ix,iz) = \frac{2\Delta t\lambda(ix,iz)}{\Delta z[2+\Delta t\kappa_z(ix,iz)]} \\
B_1^{\tau_z}(ix,iz) = \frac{2\Delta t\lambda(ix,iz)}{\Delta x[2+\Delta t\kappa_z(ix,iz)]} \\
B_2^{\tau_z}(ix,iz) = \frac{2\Delta t[\lambda(ix,iz)+2\mu(ix,iz)]}{\Delta z[2+\Delta t\kappa_z(ix,iz)]} \\
B_1^{\tau_s}(ix+0.5,iz+0.5) = \frac{2\Delta t\mu(ix+0.5,iz+0.5)}{\Delta x[2+\Delta t\kappa_z(ix+0.5,iz+0.5)]} \\
\\
B_2^{\tau_s}(ix+0.5,iz+0.5) = \frac{2\Delta t\mu(ix+0.5,iz+0.5)}{\Delta z[2+\Delta t\kappa_z(ix+0.5,iz+0.5)]}
\end{cases} \tag{A.22}$$

APPENDIX B

Software of least-squares migration

B.1 The Software of Least-squares Migration in Julia and C Languages

This appendix introduces the computer software developed for this research. The software is developed on a platform named SeismicJulia. In SeismicJulia, the software mainly integrates two essential packages: DW and LSRTM. DW package is introduced in table B.1, which generates synthetic data and wavefields for both acoustic and elastic media. Table B.2 explains the package LSRTM, which is used to solve least-square migration of both acoustic and elastic. For the sake of computational speed, wave simulations are developed in C language, which is suitable for executing codes composed of a significant amount of *for* loops. Computational environments are specified in the following tables. Moreover, the simulations are paralleled on a shared memory machine with respect to shot index.

Function	Language	**Description**
read_param_file	Julia	The function reads parameters required for wave simulation from a script file. The parameter includes spatial step-length (dx & dz), temporal step-length (dt), model size (nz, nx & nt), number of shots (ns), the filename of shot locations, number of receivers (nr), PML thickness (L), finite difference order (order).
HFD_coeff_1D	Julia	Calculate finite difference coefficients based on the required finite difference order.
PML	Julia	Calculate PML damping coefficient in the PML damping zones (top, bottom, left and right).
ACMS	Julia	The main function of Acoustic wave simulation. It runs on the master core to setup parallel framework. ACMS distributes the simulation variables and parameters to each core and then, execute the function ACM at each core.
ACM	Julia	ACM is a function used as application programming interface (API) between Julia and C language. It receives simulation variable and parameters distributed from the master core (from ACMS function), passes the information to C language and call simulation function in C language. In the C language, acoustic simulation functions are available including TWES1, TWES1q and TWES2.
TWES1	C	TWES1 solves acoustic two-way wave equations. The input is the source function. The output is data.
TWES1q	C	TWES1 computes acoustic preconditioning operator (the source-side illumination compensation). The input is the source function. The output is the preconditioning operator.
TWES2	C	TWES2 computes the acoustic source-side wavefield only. The input is the source function. The output is wavefield.
ELMS	Julia	The main function of elastic wave simulation. It runs on the master core to setup parallel framework. ELMS distributes the simulation variables and parameters to each core and then, execute the function ELM at each core.
ELM	Julia	ELM is a function used as application programming interface (API) between Julia and C language. It receives simulation variable and parameters distributed from the master core (from ACMS function), passes the information to C language and call simulation function in C language. In the C language, elastic simulation functions are available including eTWES1 and eTWES1q and eTWES2.
eTWES1	C	eTWES1 solves elastic two-way wave equations. The input is the source function. The output is data.
eTWES1q	C	eTWES1q computes elastic preconditioning operator (the source-side illumination compensation). The input is the source function. The output is the preconditioning operator.
eTWES2	C	eTWES2 computes the elastic source-side wavefield only. The input is the source function. The output is wavefield.

Table B.1: Description of functions in the DW package.

Function	Language	**Description**
read_param_file	Julia	The function reads parameters required for wave simulation from a script file. The parameter includes spatial step-length (dx & dz), temporal step-length (dt), model size (nz, nx & nt), number of shots (ns), the filename of shot locations, number of receivers (nr), PML thickness (L),finite difference order (order).
LSRTM_op	Julia	LSRTM_op is the main function to initiate parameters in least-squares and migration. The main function calls the forward operator (function AS or eAS) and adjoint operator (ATS or eATS).
AS	Julia	AS is the forward operator in acoustic least-squares migration working on the master core. It distributes inputs parameters and variables to each core and then, call A function.
A	Julia	A is a function used as application programming interface (API) between Julia and C language. It receives variable and parameters distributed from the master core (from AS function), passes the information to C language and call the forward operator (function RTM3T) in C language.
ATS	Julia	ATS is the adjoint operator in acoustic least-squares migration working on the master core. It distributes inputs parameters and variables to each core and then, call A function.
AT	Julia	A is a function used as application programming interface (API) between Julia and C language. It receives variable and parameters distributed from the master core (from AS function), passes the information to C language and call the adjoint operator (function RTM3) in C language.
eAS	Julia	eAS is the forward operator in elastic least-squares migration working on the master core. It distributes inputs parameters and variables to each core and then, call eA function.
eA	Julia	eA is a function used as application programming interface (API) between Julia and C language. It receives variable and parameters distributed from the master core (from eAS function), passes the information to C language and call the forward operator (function eRTM3T) in C language.
eATS	Julia	eATS is the adjoint operator in elastic least-squares migration working on the master core. It distributes inputs parameters and variables to each core and then, call eAT function.
eAT	Julia	eAT is a function used as application programming interface (API) between Julia and C language. It receives variable and parameters distributed from the master core (from eAS function), passes the information to C language and call the adjoint operator (function eRTM3) in C language.
RTMS	Julia	RTMS is acoustic reverse time migration operator working on the master core. ac It distributes inputs parameters and variables to each core and then, call RTM function.
RTM	Julia	RTM is a function used as application programming interface (API) between Julia and C language. It receives variable and parameters distributed from the master core (from RTMS function), passes the information to C language and call the reverse time migration operator (function RTM1) in C language.
RTM1	C	Classic reverse time migration.
RTM3	C	Acoustic exact adjoint operator.
RTM3T	C	Acoustic forward operator.
eRTM3	C	Elastic exact adjoint operator.
eRTM3T	C	Elastic forward operator.

Table B.2: Description of functions in the LSRTM package.