A GPU IMPLEMENTATION OF 2D ACOUSTIC LEAST-SQUARES REVERSE TIME MIGRATION

PHUDIT SOMBUTSIRINUN

**A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE (PHYSICS)
FACULTY OF GRADUATE STUDIES
MAHIDOL UNIVERSITY
2021**

COPYRIGHT OF MAHIDOL UNIVERSITY

Thesis
entitled

A GPU IMPLEMENTATION OF 2D ACOUSTIC LEAST-SQUARES REVERSE TIME MIGRATION

....
Mr. Phudit Sombutsirinun
Candidate

....
Chaiwoot Boonyasiriwat,
Ph.D. (Computing)
Major advisor

....
Sutthipong Noisagool,
Ph.D. (Physics)
Co-advisor

....
Prof. Patcharee Lertrit,
M.D., Ph.D. (Biochemistry)
Dean
Faculty of Graduate Studies
Mahidol University

....
Assoc. Prof. Kittiwit Matan,
Ph.D. (Physics)
Program Director
Master of Science Program in Physics
(International Program)
Faculty of Science
Mahidol University

Thesis
entitled

A GPU IMPLEMENTATION OF 2D ACOUSTIC LEAST-SQUARES REVERSE TIME MIGRATION

was submitted to the Faculty of Graduate Studies, Mahidol University
for the degree of Master of Science (Physics)
on
March 24, 2021

....
Mr. Phudit Sombutsirinun
Candidate

....
Asst. Prof. Siriporn Chaisri,
Ph.D. (Geophysics)
Chair

....
Chaiwoot Boonyasiriwat,
Ph.D. (Computing)
Member

....
Sutthipong Noisagool,
Ph.D. (Physics)
Member

....
Prof. Patcharee Lertrit,
M.D., Ph.D. (Biochemistry)
Dean
Faculty of Graduate Studies
Mahidol University

....
Assoc. Prof. Palangpon Kongsaeree,
Ph.D. (Chemistry)
Dean
Faculty of Science
Mahidol University

ACKNOWLEDGEMENTS

This thesis has been proceeded under the guidance of Dr. Chaiwoot Boonyasiriwat of Mahidol University Center of Scientific Computing (MCSC), and Dr. Sutthipong Noisagool of Geophysics laboratory, Department of Physics, Faculty of Science, Mahidol University.

Without many supports that I have received, this work would not be completed. Therefore, I would like to acknowledge every assistance in this section. First of all, I would like to express my deepest gratitude to Dr. Chaiwoot Boonyasiriwat, my major advisor, for the suggestion of thesis topic, essential lectures, and important advices along the process of this thesis. I would also like to thank Dr. Sutthipong Noisagool for being the co-advisor, and being a great brother for me to discuss on every topics. With all of my gratitude, I would like to thank Asst. Prof. Dr. Siriporn Chaisri, my advisor during my undergraduate study and the chair of this thesis defense, those advices that I got during those days have forged me to become who I am today. Indispensably, every body at MCSC and Geophysics laboratory for making such a good working environment. The scholarship for young scientist and the financial support for being a teacher assistant from Faculty of Science, Mahidol University, help me settled in a lot easier.

At last, I would like to thank my family and also Bonut for being with me on both good and bad times, assisting me through out this journey. I love you all.

Phudit Sombutsirinun

A GPU IMPLEMENTATION OF 2D ACOUSTIC LEAST-SQUARES REVERSE TIME MIGRATION

PHUDIT SOMBUTSIRINUN 6038186 SCPY/M

M.Sc. (PHYSICS)

THESIS ADVISORY COMMITTEE: CHAIWOOT BOONYASIRIWAT, Ph.D., SUTTHIPONG NOISAGOOL, Ph.D.

ABSTRACT

Least-squares reverse time migration (LSRTM) is a seismic imaging method that can provide higher-resolution images of the subsurface structures compared to other methods. However, LSRTM is computationally expensive. To reduce the computational time of LSRTM, GPU can be utilized. This leads to the objective of this work which is to develop a GPU implementation of LSRTM. In this work, the two-dimensional first-order acoustic wave equations were solved using the second-order finite difference method on a staggered grid, and a perfectly matched layer was used as an absorbing boundary condition. The adjoint-state method was used to compute the gradient of the objective function concerning model parameters. A linear conjugate gradient method was used to minimize the objective function. Both forward- and backward-propagation of wavefields using the finite-difference method were performed on a single GPU using the NVIDIA CUDA library. For verification purposes, the GPU program of LSRTM was applied to a synthetic data set generated from the Marmousi model. Numerical results showed that LSRTM could provide an image with higher quality compared to a conventional RTM image. The GPU-version of LSRTM has acquired a speedup factor of 12-to-13 times compared to the serial CPU-version of LSRTM.

KEYWORDS : SEISMIC IMAGING/ GPU IMPLEMENTATION/ CUDA/
LEAST-SQUARES REVERSE TIME MIGRATION

67 pages

CONTENTS

Page

ACKNOWLEDGEMENTS **iii**

ABSTRACT **iv**

LIST OF FIGURES **vii**

CHAPTER I INTRODUCTION **1**

1.1 Introduction 1

1.2 Objective 6

1.3 Thesis Outline 6

CHAPTER II METHODS **7**

2.1 First-Order Coupled Acoustic Wave Equations 7

2.2 Perfectly-Matched Layer 8

2.3 Solving Acoustic Wave Equation by Finite Difference Method Staggered Grid 9

2.4 Acoustic Least-Squares Reverse Time Migration 12

2.5 Gradient Computation using the Adjoint-State Method 14

CHAPTER III IMPLEMENTATION **19**

3.1 GPU Implementation 19

3.2 PS_LSRTM Algorithm 24

3.3 Model Parameterization 25

3.4 Generate Source 26

3.5 RTM Variables 27

3.6 Generate Seismogram 28

3.7 RTM Method 29

3.8 LSRTM Variables 31

3.9 Migrated Image Modeling 31

3.10 Gradient of Objective Function Computation 32

3.11 Conjugate Gradient Method 33

CONTENTS (cont.)

Page

CHAPTER IV NUMERICAL RESULTS 35

4.1 Wavefield Reconstruction 35

4.2 Two-Horizontal Layered Model 37

4.3 Marmousi Model 46

4.4 Speedup Factor 59

CHAPTER V CONCLUSION 61

5.1 Conclusion 61

5.2 Possible Improvement 61

BIOGRAPHY 67

LIST OF FIGURES

Figure **Page**

1.1 A schematic of RTM, adapted Etgen et al. (2009) and Liu et al. (2011). 2

1.2 A migration image from RTM which corrupted by low-frequency noise from Guitton et al. (2007) 3

1.3 Numerical results from Bourgeois et al. (1989) including the test model (left), time reversal migration (upper right), and linearized inversion image (lower right). 3

1.4 A 2-D sketch of required points for boundary saving for regular grid finite difference of Dussaud et al. (2008) scheme for eighth-order finite difference stencil from Yang et al. (2014) 4

1.5 16×16 data tiles and halos for 8^{th}-order FD stencil from Micikevicius (2009) 5

2.1 Variables storage in 2D staggered FD grid, where i and j are indices for horizontal and vertical direction, respectively. A blue circle reprensents pressure wavefield and bulk modulus, while green and orange triangel represent particle motion velocity and the inverse of density. 10

2.2 Spatial stencil in 2D staggered FD grid for the first-order coupled acoustic wave equation. 10

2.3 Advance time stepping in staggered FD grid, where k is a time index. 11

2.4 Flowchart of LSRTM method, where it is iteration index. 18

3.1 CPU and GPU architecture (NVIDIA, 2018) 19

3.2 Grid of blocks and block of threads (NVIDIA, 2018) 20

3.3 An example code of adding matrices in CUDA (NVIDIA, 2018) 21

3.4 Sequential execution of heterogenous programming with the detail of CUDA built-in variables and function usage (NVIDIA, 2018). 22

3.5 8^{th}-order 3D FD Stencil, where red cube represents the target node (NVIDIA, 2018) 23

LIST OF FIGURES (cont.)

Figure **Page**

3.6 Shared memory used in FD stencil for staggered grid of the first-order coupled acoustic wave equation 23

3.7 PS_LSRTM workflow 25

3.8 2D Guassian distribution ($\sigma = 0.2$) 26

3.9 Ricker wavelet ($f = 5\ Hz$) 27

3.10 A flowchart of the forward modeling scheme 29

3.11 A flowchart of the RTM method 30

3.12 The migrated image modeling flowchart 32

3.13 The gradient of objective function computation flowchart 33

3.14 Conjugate gradient method flowchart 34

4.1 Wavefield propagation in a homogeneous medium (a) shows a forward propagation at 0.16617 s and (b) at 0.22156 s. A backward propagation at 0.22156 s and 0.16617 s are displayed at (d) and (c), respectively. A red square line indicate the physical region. 36

4.2 P-wave velocity model (left) and smooth P-wave velocity model (right) of the two-horizontal layered model 37

4.3 Seismogram of two-layered model with label of seismic events, D for direct wave, R for reflect wave 38

4.4 Migrated images from the LSRTM method on two-layered model with 1 20-Hz source 39

4.5 Residual graph of LSRTM on two-layered model with 1 20-Hz source 39

4.6 Migrated images from the LSRTM method on two-layered model with 10 20-Hz sources 40

4.7 Residual graph of LSRTM on two-layered model with 10 20-Hz sources 41

4.8 Migrated images from the LSRTM method (RTM image from residual data) on two-layered model with 10 20-Hz sources 41

LIST OF FIGURES (cont.)

Figure **Page**

4.9 Residual graph of LSRTM (RTM image from residual data) on two-layered model with 10 20-Hz sources 42

4.10 Procedure to mute the direct wave from seismogram of two-layered model 43

4.11 Migrated images from the LSRTM method (muted direct wave) on two-layered model with 10 20-Hz sources 44

4.12 Residual graph of LSRTM (muted direct wave) on two-layered model with 10 20-Hz sources 44

4.13 Migrated images from the LSRTM method (muted surface) on two-layered model with 10 20-Hz sources 45

4.14 Residual graph of LSRTM (muted surface) on two-layered model with 10 20-Hz sources 45

4.15 P-wave velocity model (left) and smooth P-wave velocity model (right) of Marmousi model 46

4.16 Seismogram of Marmousi model 47

4.17 Migrated images from the LSRTM method on Marmousi model with 31 5-Hz sources 48

4.18 Residual graph of LSRTM on Marmousi model with 31 5-Hz sources 48

4.19 Migrated images from the LSRTM method (RTM image from residual data) on Marmousi model with 31 5-Hz sources 49

4.20 Residual graph of LSRTM method (RTM image from residual data) on Marmousi model with 31 5-Hz sources 49

4.21 Procedure to mute the direct wave from seismogram of Marmousi model 50

4.22 Migrated images from the LSRTM method (muted direct wave) on Marmousi model with 31 5-Hz sources 51

4.23 Residual graph of the LSRTM method (muted direct wave) on Marmousi model with 31 5-Hz sources 51

LIST OF FIGURES (cont.)

Figure **Page**

4.24 Migrated images from the LSRTM method on Marmousi model with 101 5-Hz sources 52

4.25 Migrated images with differential filter from the LSRTM method on Marmousi model with 101 5-Hz sources 53

4.26 Residual graph of the LSRTM method on Marmousi model with 101 5-Hz sources) 53

4.27 A change of migrated image during the inversion scheme 54

4.28 Migrated images from the LSRTM method on Marmousi model with 31 10-Hz sources 55

4.29 Migrated images with differential filter from the LSRTM method on Marmousi model with 31 10-Hz sources 55

4.30 Residual graph of the LSRTM method on Marmousi model with 31 10-Hz sources 56

4.31 Migrated images from the LSRTM method on Marmousi model with 71 10-Hz sources 57

4.32 Migrated images with differential filter from the LSRTM method on Marmousi model with 71 10-Hz sources 57

4.33 Residual graph of the LSRTM method on Marmousi model with 71 10-Hz sources 58

4.34 Residual data of the Marmousi model with 71 10-Hz sources in each iteration 58

4.35 The runtime of FD calculation with varied numbers of data of CPU (Orange line) and GPU (Blue line) with the speedup factor in each problem size (Yellow line) 59

4.36 The runtime of LSRTM metohd with varied numbers of data of CPU (Orange line) and GPU (Blue line) with the speedup factor in each problem size (Yellow line) 60

LIST OF FIGURES (cont.)

Figure **Page**

5.1 RTM image with (a) conventional imaging condition and (b) Liu et al. (2011) imaging condition from Liu et al. (2011) 62

5.2 Performance metrics showing the mean of ten trials for various cube (N^3) model domains using the *ewefd3d* code. (a) Computational run time for CPU (green line), a single GPU (blue line), two GPUs with MPI communication (red line), and two GPU with P2P communication (magenta line). (b) Speedup relative to CPU for a single GPU (red line), and two GPUs with MPI (blue line) and P2P communication (magenta line). (c) Relative speed for the P2P versus MPI communication. 63

CHAPTER I
INTRODUCTION

1.1 Introduction

The ultimate goal of seismic exploration is to determine the location of oil and gas reservoirs. To be able to accurately locate the reservoir, a transformation of a seismic shot record into a geologic structure is needed and that tool is named as seismic imaging. The most widely used seismic imaging technique is reflection seismic imaging in which reflection data are used to construct an image of the subsurface structures. In reflection seismic imaging, three main methods to image a geologic model are Kirchoff migration, one-way wave equation migration, and reverse time migration (RTM). Kirchoff migration is a ray-based method that efficiently image steeply dipping structure, but this method has its own limit since the ray-based algorithm always face the shadow zone which is an area that the rays can not reach. Therefore, the complex geologic structure can not be fully delineated by Kirchoff migration. To deal with the complex structure, the one-way wave equation migration is presented. This method uses the paraxial approximation to propagate a wavefield, and the shadow zone is not existed since this is the wave-equation based method. However, the one-way wave equation migration is restricted by the shooting angle of the wavefield. More detailed analysis of Kirchhoff and one-way wave migration can be found in Etgen et al. (2009). On the other hand, RTM algorithm utilizes two-way wave equation, neither shadow-zoned nor steeply dipping reflector problem are exists. Nevertheless, RTM is a depth migration that needs an accurate velocity model to heighten its efficiency (Chang and McMechan, 1986). The RTM method consists of three steps. First, the wavefield is propagated forward in time from source through the velocity model to the receivers. This wavefield can be named as source wavefield which will be used later on this scheme. The simulation of source wavefield also grants another product which is a seismogram or seismic data. Second, the seismogram will be the source of receivers wavefield that propagates backward in time. Lastly, a zero-lagged crosscorrelation of source and receiver wavefields yields a reflectivity-like image, since the point that source and receiver wavefields share the same path at the same time is the point of the reflector. However, using the two-way wave equation grants an ability to simulate both transmitted and reflected wave. With this condition, the crosscorrelation may not correspond to reflection from a reflector. This problem leads to the crosstalk noise in the RTM image. A schematic of RTM is picturized in Figure 1.1.

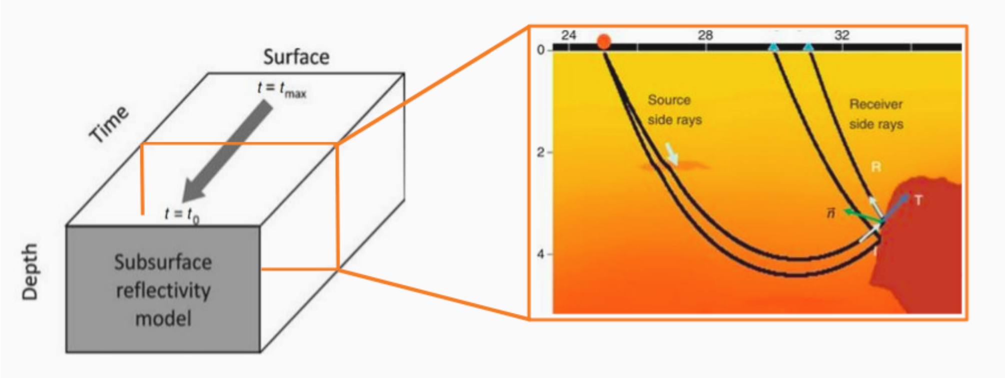


Figure 1.1: A schematic of RTM, adapted Etgen et al. (2009) and Liu et al. (2011).

The RTM method was first proposed as a post-stack migration method by Baysal et al. (1983) and McMechan (1983). At that time, the algorithm of the post-stack migration is just a performing of a finite-difference calculation backward in time. Later then, Chang and McMechan (1986) proposed a pre-stack RTM method which applies an imaging condition or the crosscorrelation of source and receiver wavefields to produce the image (Claerbout, 1971). As stated that RTM needs the accurate velocity model, its resolution is also heavily affected by the complexity of a model. The RTM images are normally corrupted with low frequency noise produced from the sharp velocity contrast in the model and the unwanted crosscorrelation as shown in Figure 1.2 (Guitton et al., 2007).

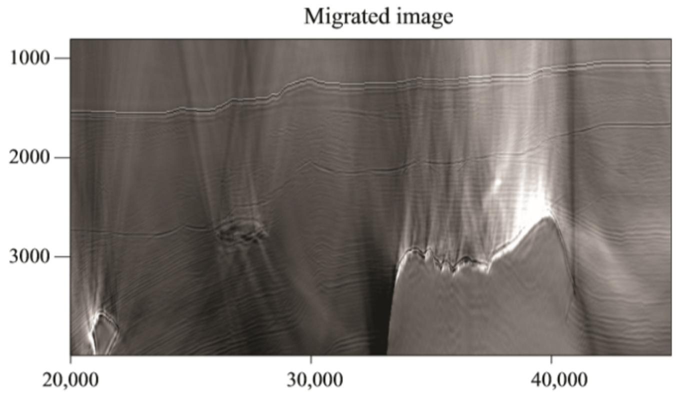


Figure 1.2: A migration image from RTM which corrupted by low-frequency noise from Guitton et al. (2007)

One of the method that can improve the RTM image quality is the least-squares reverse time migration (LSRTM) method. As its name stated, LSRTM is the least-squares inversion implemented on RTM. Plessix (2006) exhibited how the gradient of objective function can be obtained by the crosscorrelation of forward- and backward- propagating wavefields. Hence, LSRTM is literally the execution of RTM iteratively which should improve the result. LSRTM has been proposed since Bourgeois et al. (1989) under the name of a linearized inversion method which can be seen as an improvement of the pre-stack RTM of Chang and McMechan (1986). Results from Bourgeois et al. (1989)'s work are shown in Figure 1.3.

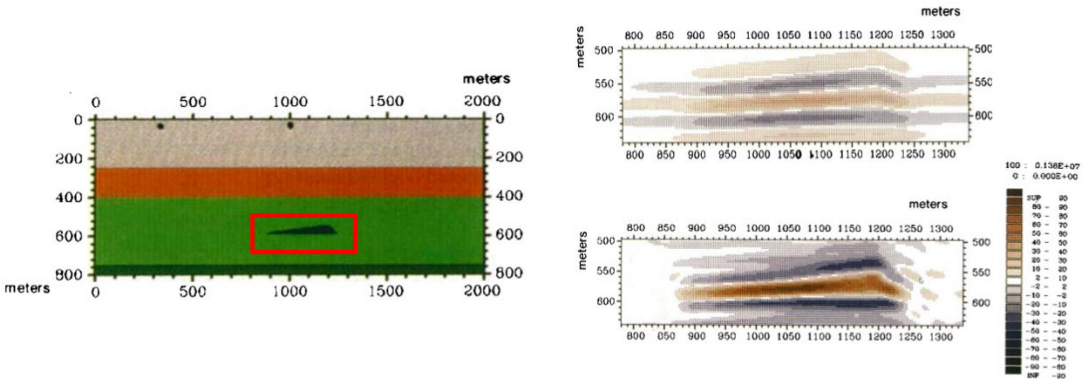


Figure 1.3: Numerical results from Bourgeois et al. (1989) including the test model (left), time reversal migration (upper right), and linearized inversion image (lower right).

LSRTM also reduces the noise from acquisition footprint and RTM artifacts (Dai and

Schuster, 2013). Despite the better resolution of the image, LSRTM was not commonly used due to its high requirement of computational storage and time (Dussaud et al., 2008). Since, RTM algorithm needs the vast storage for storing every time slice of the source wavefield in order to crosscorrelate with the receivers wavefield in the imaging condition step. Dussaud et al. (2008) described and analyzed the existing methods to solve this problem, and a widely used approach is the wavefield reconstruction. The wavefield reconstruction for dissipative boundary condition requires the storage of wavefield at the boundary at every timestep. These boundary wavefields will be used as initial data to reconstruct the source wavefield backward in time. An example of boundary saving is shown in Yang et al. (2014)'s work for the eighth-order finite difference stencil in Figure 1.4. There is also another factor that affects the migration image. Feng and Schuster (2017) demonstrated that their elastic LSRTM image has a better quality than acoustic LSRTM. Furthermore, Dutta and Schuster (2014) also shown that the convergence of the objective function can be quickening by using visco-acoustic LSRTM.

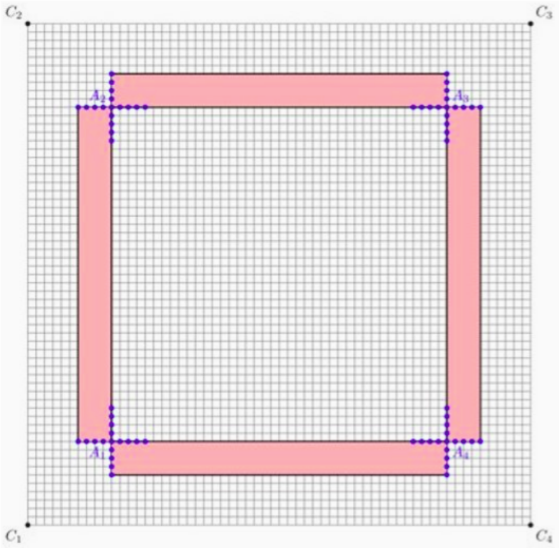


Figure 1.4: A 2-D sketch of required points for boundary saving for regular grid finite difference of Dussaud et al. (2008) scheme for eighth-order finite difference stencil from Yang et al. (2014)

For the computational cost issue of LSRTM, the GPU programming is the main solution due to GPU ability to efficiently process many data at the same time. A GPU implementation of seismic imaging gained its popularity since the introduction of finite-difference (FD) calculation on GPUs using CUDA by Micikevicius (2009). The main idea of Micikevicius (2009) is to utilize the shared memory to store the data tiles along with the halo nodes. The halo nodes or halos are FD stencil of the boundary data on the data tile. An example of data tiles and halos is shown in Figure 1.5.

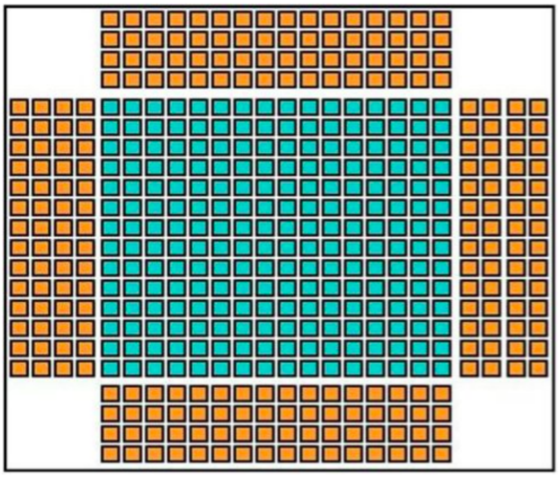


Figure 1.5: 16×16 data tiles and halos for 8^{th}-order FD stencil from Micikevicius (2009)

Later, Abdelkhalek et al. (2009) implemented Micikevicius (2009) method on RTM and achieved 30- and 10-times speedup for modeling and RTM schemes, respectively. Abdelkhalek et al. (2009) also stated a major problem of multiple GPUs programming which is a bottle neck problem. Due to a low transfer rate of GPU, the speedup ratio between CPU and GPU gets lower with more domain decomposition. There are also published software packages that also boost up the popularity of GPU implementation in seismic methodology. For instance, Weiss and Shragge (2013) who also adopted Micikevicius (2009) method and published 2D and 3D GPU-based anisotropic elastic modeling code. Weiss and Shragge (2013)'s $ewef2d$ grants 10 times speedup, whereas $ewefd3d$ yields 16-times speedup for single GPU and 28-times speedup for two GPUs. This work also suggested the CUDA's P2P communication between multiple GPUs within the same node over the MPI-based communication. Yang (2015) published a 2D time-domain acoustic full-waveform inversion (FWI). Yang (2015) attemped to improve an efficiency of an optimization scheme with paralled reduction, but the acquired speedup ratio was hindered by the effect of FD calculation.

1.2 Objective

In the best of my knowledge, there is no open-source GPU implementation of LSRTM. Therefore, the objective of this work is to implement 2D acoustic LSRTM on a GPU using CUDA and publish it as an open-source software. In Yang (2015)'s FWI open-source code, a regular FD grid and the zero-degree Clayton-Enquist absorbing boundary condition were used. On the contrary, my open-source code will use a staggered FD grid and perfectly matched layers (PML) since the staggered grid is more efficient than the regular grid and PML is more efficient and more effective than the Clayton-Enquist absorbing boundary condtion.

1.3 Thesis Outline

This thesis is divided into 5 chapters. After outlining the entire thesis in the first chapter, every methods used in LSRTM are provided in Chapter 2. The simulation of wavefield propagation through the medium by the first-order coupled acoustic wave equations along with the finite difference method are described first in Chapter 2. Then, the gradient computation, the inversion scheme of LSRTM method also explained. The developed GPU-based LSRTM source code, named PS_LSRTM, and GPU implementation are presented in Chapter 3. A flowchart of the PS_LSRTM algorithm is presented and each part of the algorithm is fully explained. The numerical results from this source code such as seismic shot record, migrated image from RTM method, optimized migrated image from LSRTM method, convergent rate graph, and speedup ratio are shown and discussed in Chapter 4 along with the model and acquisition parameters. Lastly, the thorough content of this thesis will be summed up in Chapter 5.

CHAPTER II
METHODS

In this work, seismic wave propagation is approximately governed by the first-order coupled acoustic wave equations. Therefore, the simulation of acoustic wave by finite difference method will be stated in this chapter first. Then, LSRTM method will be furthur explained in detail.

2.1 First-Order Coupled Acoustic Wave Equations

Acoustic wave is a mechanical wave which needs a medium to propagate. A change of pressure field in time can be affected by the medium properties such as bulk modulus, density, and a change of particle motion velocity and vice versa. Hence, the two-dimensional first-order coupled acoustic wave equations can be expressed as

$$\frac{\partial p}{\partial t} = -K\left(\frac{\partial v_x}{\partial x} + \frac{\partial v_z}{\partial z}\right) \tag{2.1}$$

$$\frac{\partial v_x}{\partial t} = -\frac{1}{\rho}\frac{\partial p}{\partial x} \tag{2.2}$$

$$\frac{\partial v_z}{\partial t} = -\frac{1}{\rho}\frac{\partial p}{\partial z} \tag{2.3}$$

where p is pressure wavefield, v_x and v_z are particle motion velocity with x and z subscript stand for horizontal and vertical direction, respectively. K and ρ represent bulk modulus and density of the medium, while the wave velocity in medium (c) can be calculated from $c = \sqrt{K/\rho}$. The above equations are an acoustic approximation of P-SV equation of motions in 2D from Levander (1988), and the similar form of these first-order couple acoustic wave equations can be seen from the work of Zhou (2004).

2.2 Perfectly-Matched Layer

By solving acoustic wave equation, a propagation of wave or wave modeling can be achieved. Nevertheless, an actual wave propagate in the boundless area which can not be achieved by numerical modeling due to the limit of problem size. The solution to this problem is the artificial layer to absorb the wave energy or perfectly-matched layer (PML). PML is first proposed by Berenger (1994) and it becomes the most widely used boundary condition for modeling in an unbounded domain. The significant part of the PML method is the PML damping function ($\sigma(x)$), which is the function of depth that defines how much the wave amplitude will be damped at the boundary. According to Colino and Tsogka (2001), the PML damping function is

$$\sigma(x) = \sigma_0 \left(\frac{x}{\delta}\right)^2 \tag{2.4}$$

$$\sigma_0 = log\left(\frac{1}{R}\right)\frac{3v}{2\delta} \tag{2.5}$$

R is a reflectivity coefficient that usually have value around 10^{-7} to 10^{-4} and δ is a PML thickness. The next step is to apply the coordinate stretching, which is

$$\frac{\partial}{\partial\hat{x}} = \frac{1}{1+\frac{i\sigma(x)}{\omega}}\frac{\partial}{\partial x} \tag{2.6}$$

As equation (2.5) and (2.6) show, the implementation is one-directional. Hence, the pressure wavefield in equation (2.1) is needed to be split into

$$\frac{\partial p_x}{\partial t} = -K\left(\frac{\partial v_x}{\partial x}\right) \tag{2.7}$$

$$\frac{\partial p_z}{\partial t} = -K\left(\frac{\partial v_z}{\partial z}\right) \tag{2.8}$$

where $p = p_x + p_z$. Moreover, the stretched coordinate is in a frequency domain, so the wave equations must be transformed into the frequency domain as well. Henceforth, equations (2.7), (2.8), (2.2), and (2.3) will go through the Fourier transformation

$$-i\omega\hat{p}_x = -K\left(\frac{\partial\hat{v}_x}{\partial x}\right) \tag{2.9}$$

$$-i\omega\hat{p}_z = -K\left(\frac{\partial\hat{v}_z}{\partial z}\right) \tag{2.10}$$

$$-i\omega\hat{v}_x = -\frac{1}{\rho}\frac{\partial\hat{p}}{\partial x} \tag{2.11}$$

$$-i\omega\hat{v}_z = -\frac{1}{\rho}\frac{\partial\hat{p}}{\partial z} \tag{2.12}$$

Plug the stretched coordinate (equation (2.6)) into equations (2.9), (2.10), (2.11), and (2.12). With an arrangement and conversion back to time domain into these equations, the split-field acoustic wave equations with PML boundary are

$$\frac{\partial p_x}{\partial t} + \sigma(x)p_x = -K\frac{\partial v_x}{\partial x} \tag{2.13}$$

$$\frac{\partial p_z}{\partial t} + \sigma(z)p_z = -K\frac{\partial v_z}{\partial z} \tag{2.14}$$

$$\frac{\partial v_x}{\partial t} + \sigma(x)v_x = -\frac{1}{\rho}\frac{\partial p}{\partial x} \tag{2.15}$$

$$\frac{\partial v_z}{\partial t} + \sigma(z)v_z = -\frac{1}{\rho}\frac{\partial p}{\partial z} \tag{2.16}$$

Equations (2.13), (2.14), (2.15), and (2.16) will be solved by finite difference method to resemble the wave propagation in the unbounded domain.

2.3 Solving Acoustic Wave Equation by Finite Difference Method Staggered Grid

To be able to simulate the wavefield by finite difference method, the geologic model is divided into mesh with grid size h in x and z direction. The model parameters and modeling variables are stored in each node in the model. How those variables are stored will tell a type of grid using in modeling. In regular grid, every variables are stored on the same node. On the other hand, the variables are stored on node and between node in staggered grid which can be seen in Figure 2.1 with indices i and j represent x and z, respectively.

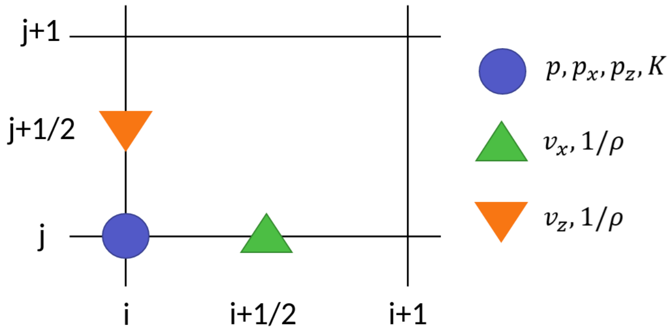


Figure 2.1: Variables storage in 2D staggered FD grid, where i and j are indices for horizontal and vertical direction, respectively. A blue circle reprensents pressure wavefield and bulk modulus, while green and orange triangel represent particle motion velocity and the inverse of density.

With first-order coupled equations being used, staggered grids are suitable due to stability (Virieux, 1986). Furthermore, staggered grids tend to have more potential in avoiding odd-even decoupling that can lead to checker board pattern in the numerical solution (Neill and Hashemi, 2018). The staggered FD grid solves a spatial and temportal derivatives with the pattern shown in Figure 2.2 and 2.3.

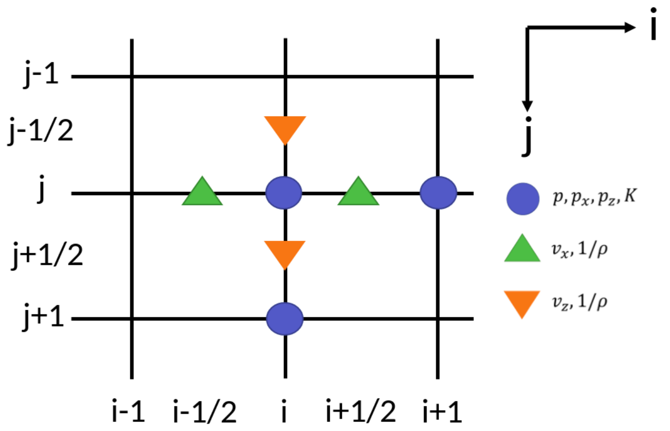


Figure 2.2: Spatial stencil in 2D staggered FD grid for the first-order coupled acoustic wave equation.

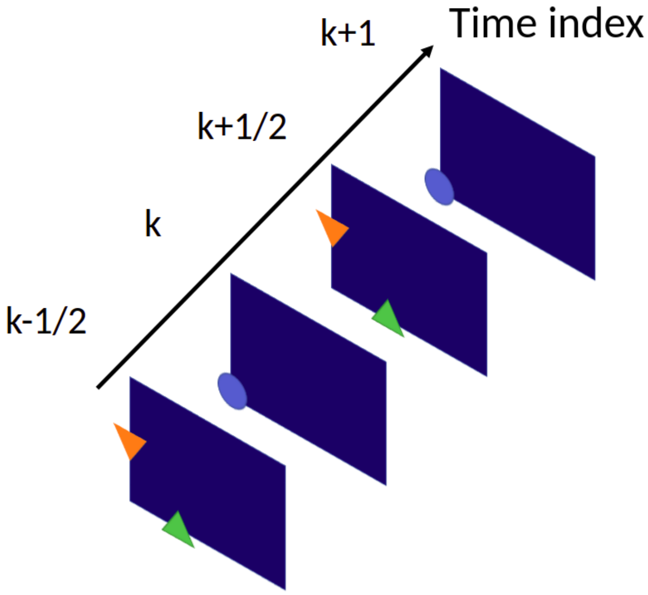


Figure 2.3: Advance time stepping in staggered FD grid, where k is a time index.

Figure 2.2 shows the second-order FD order for the first-order coupled acoustic wave equations. A value of v_x on coordinate $(i+1/2, j)$ is solved by the central FD calculation of p at $(i+1, j)$ and (i, j). Similarly, the value of v_z on coordinate $(i-1/2, j)$ is solved by the value of p at $(i, j+1)$ and (i, j). For the FD calculation of p_x and p_z at (i, j), the adjacent nodes of v_x and v_z are used, respectively. The spatial FD calculation of p at time index k together with the value of v_x and v_z at time index $k-1/2$ are used to advance the time step of v_x and v_z to the time index $k+1/2$. Similarly, the advance time stepping of p from time index k to $k+1$ used the FD calculation of v_x and v_z at time index $k+1/2$. This procedure is depicted in Figure 2.3. Putting the FD stencil and advance time stepping scheme to equations eqrefpmlpx, (2.14), (2.15), and (2.16) yield the discrete form of the first-order coupled acoustic wave equation with PML boundary.

$$\frac{p_{x_{i,j}}^{k+1} - p_{x_{i,j}}^{k}}{\Delta t} + \sigma_{i,j} p_{x_{i,j}}^{k} = -K_{i,j} \left(\frac{v_{x_{i+1/2,j}}^{k+1/2} - v_{x_{i-1/2,j}}^{k+1/2}}{\Delta x} \right) \tag{2.17}$$

$$\frac{p_{z_{i,j}}^{k+1} - p_{z_{i,j}}^{k}}{\Delta t} + \sigma_{i,j} p_{z_{i,j}}^{k} = -K_{i,j} \left(\frac{v_{z_{i,j+1/2}}^{k+1/2} - v_{z_{i,j+1/2}}^{k+1/2}}{\Delta z} \right) \tag{2.18}$$

$$\frac{v_{x_{i,j}}^{k+1/2} - v_{x_{i,j}}^{k-1/2}}{\Delta t} + \sigma_{i,j} v_{x_{i,j}}^{k-1/2} = -\left(\frac{b_{i,j} + b_{i+1,j}}{2} \right) \left(\frac{p_{i+1,j}^{k} - p_{i,j}^{k}}{\Delta x} \right) \tag{2.19}$$

$$\frac{v_{z_{i,j}}^{k+1/2} - v_{z_{i,j}}^{k-1/2}}{\Delta t} + \sigma_{i,j} v_{z_{i,j}}^{k-1/2} = -\left(\frac{b_{i,j} + b_{i,j+1}}{2} \right) \left(\frac{p_{i,j+1}^{k} - p_{i,j}^{k}}{\Delta z} \right) \tag{2.20}$$

Where b is buoyancy or the inverse of density ρ, and k stands for time index t.

2.4 Acoustic Least-Squares Reverse Time Migration

The forward modeling scheme which perform a modeling to the model to achieve the seismogram can be written in matrix form as

$$\mathbf{D} = \mathbf{L}\mathbf{m} \tag{2.21}$$

where **L** is the forward modeling operator. The product of operator **L** can be differ according to an operand. If it acts on the true model, the seismogram **D** will be the result. On the other hand, operating **L** on an initial model of an inversion scheme will grant the simulated seimogram **d**. The goal of the least-squares inversion scheme is to find the model **m** that minimize the difference between these two parameters. To clarify that, the model **m** is sought to minimize the objective function

$$f(\mathbf{m}) = \frac{1}{2} \sum_{is}^{N_s} \|\mathbf{L}_{is}\mathbf{m} - \mathbf{D}_{is}\|^2 \tag{2.22}$$

where N_s is the number of sources and is is the index of source number. Operator **L** that acted on the model parameter is conventional, but, in LSRTM method, variable **m** is a migated image. To clarify this, the algorithm of modeling the image is presented.

A derivation can be done by perturbation theory. Considering the constant density problem, the model parameter can be extracted into

$$K = K_0 + \delta K \tag{2.23}$$

that produces the wavefield

$$p = p_0 + \delta p \tag{2.24}$$

$$v_x = v_{x0} + \delta v_x \tag{2.25}$$

$$v_z = v_{z0} + \delta v_z \tag{2.26}$$

The subscript 0 represents the background value of the model, while the δ represent the perturbed value. Substituting those extracted parameters into the wave equation (equation (2.1)) gives

$$\frac{\partial}{\partial t}(p_0 + \delta p) + (K_0 + \delta K)\left(\frac{\partial}{\partial x}(v_{x0} + \delta v_x) + \frac{\partial}{\partial z}(v_{z0} + \delta v_z)\right) = 0 \tag{2.27}$$

Neglect the multiplication of δK and δu and subtract equation (2.27) by equation (2.1) yields

$$\begin{aligned} \frac{\partial \delta p}{\partial t} + K_0\left(\frac{\partial \delta v_x}{\partial x} + \frac{\partial \delta v_z}{\partial z}\right) &= -\delta K\left(\frac{\partial \delta v_{x0}}{\partial x} + \frac{\partial \delta v_{z0}}{\partial z}\right) \\ &= -\frac{\delta K}{K_0}K_0\left(\frac{\partial \delta v_{x0}}{\partial x} + \frac{\partial \delta v_{z0}}{\partial z}\right) \\ &= mK_0\left(\frac{\partial \delta v_{x0}}{\partial x} + \frac{\partial \delta v_{z0}}{\partial z}\right) \end{aligned} \tag{2.28}$$

where $m = \frac{\delta K}{K_0}$ is a fractional bulk modulus that is used as the image of reflectivity. Apply this scheme to every wave equations to acquire the wave equations used for modeling the migrated image.

$$\frac{\partial \delta p}{\partial t} + K_0\left(\frac{\partial \delta v_x}{\partial x} + \frac{\partial \delta v_z}{\partial z}\right) = mK_0\left(\frac{\partial \delta v_{x0}}{\partial x} + \frac{\partial \delta v_{z0}}{\partial z}\right) \tag{2.29}$$

$$\frac{\partial \delta v_x}{\partial t} + \frac{1}{\rho}\frac{\partial \delta p}{\partial x} = 0 \tag{2.30}$$

$$\frac{\partial \delta v_z}{\partial t} + \frac{1}{\rho}\frac{\partial \delta p}{\partial z} = 0 \tag{2.31}$$

Carry on with the LSRTM algorithm, the optimization of **m** value can be done by using the gradient descent method such as conjugate gradient method (CG) which is implemented in this work. The main idea of gradient descent method is to update the model by the gradient of objective function (equation (2.22)) which can achieved by taking the derivative of **f** respects to **m**.

$$\mathbf{g}^{it+1} = \mathbf{L}^T(\mathbf{L}\mathbf{m}^{it} - \mathbf{D}) \tag{2.32}$$

where **g** is the gradient of objective function, and superscript it is used as an index of iteration. Then, CG algorithm will adjust this gradient **g** into **z** by factor β to speed up the convergent rate and use it to update the model by the factor α of **z**. The CG algorithm, following Dai and Schuster (2013) will be formulated as

$$\beta = \frac{\left[\mathbf{g}^{it+1}\right]^T \mathbf{g}^{it+1}}{\left[\mathbf{g}^{it}\right]^T \mathbf{g}^{it}} \tag{2.33}$$

$$\mathbf{z}^{it+1} = -\mathbf{g}^{it+1} + \beta \mathbf{z}^{it} \tag{2.34}$$

$$\alpha = \frac{\left[\mathbf{z}^{it+1}\right]^T \mathbf{g}^{it+1}}{\left[\mathbf{L}\mathbf{z}^{it+1}\right]^T \mathbf{L}\mathbf{z}^{it+1}} \tag{2.35}$$

$$\mathbf{m}^{it+1} = \mathbf{m}^{it} + \alpha \mathbf{z}^{it+1} \tag{2.36}$$

2.5 Gradient Computation using the Adjoint-State Method

By using the adjoint-state method, the gradient of objective function that is needed in LSRTM method can be computed by modeling forward- and backward-propagation wavefields. Apart from gradient computation that will be used in the inversion scheme, the derivation of adjoint-state method can also lead to the RTM method. Now, the adjoint-state method will be applied to the LSRTM method following the procedure presented in Plessix (2006). For simplicity, the acoustic wave equations will be in the form of

$$\frac{\partial p}{\partial t} + K \nabla \cdot \mathbf{v} = 0 \tag{2.37}$$

$$\frac{\partial \mathbf{v}}{\partial t} + \frac{1}{\rho} \nabla p = 0 \tag{2.38}$$

where **v** consists of v_x and v_z. The forward modeling can be summed up into

$$F(\mathbf{u}(\mathbf{m}), \mathbf{m}) = 0 \tag{2.39}$$

where **u** is a state variable which is wavefields variables for this occasion, and **m** is model parameters which consists only K due to constant density approximation. While, the objective function is

$$h(\mathbf{u}(\mathbf{m}), \mathbf{m}) = \frac{1}{2} \int_0^T \int (D - d)^2 dx dt \tag{2.40}$$

where T is the record length, D is the seismogram acquired from true velocity model, and d is the seismogram acquired from simulation. A reminder that seismogram is the pressure field at the receivers position at every timestep ($p(\mathbf{x}_r, t)$). Functional h and F will be used to form the Lagrangian ($\mathcal{L}$) that can be used for compute the gradient of objective function and also backward-propagation modeling.

$$\mathcal{L}(\tilde{\mathbf{u}}, \tilde{\boldsymbol{\lambda}}, \mathbf{m}) = h(\tilde{\mathbf{u}}, \mathbf{m}) + \left\langle \tilde{\boldsymbol{\lambda}}, F(\tilde{\mathbf{u}}, \mathbf{m}) \right\rangle \tag{2.41}$$

where $\boldsymbol{\lambda}$ represents the adjoint-variable which consists of the adjoint variable of p (λ_p) and $\mathbf{v}$ ($\lambda_{\mathbf{v}}$). While, a symbol tilde is identified for the benefit of derivation, and $<, >$ indicates the inner product. Substitute equations (2.37), (2.38), and (2.40) into equation (2.41) yields

$$\mathcal{L}(\tilde{u}, \tilde{\lambda}, m) = \int_0^T \int \frac{1}{2}(D-d)^2 dxdt - \int_0^T \left\langle \tilde{\lambda}_p, \frac{\partial \tilde{p}}{\partial t} + K\nabla\cdot\tilde{\mathbf{v}} \right\rangle dt - \int_0^T \left\langle \tilde{\lambda}_{\mathbf{v}}, \frac{\partial \tilde{\mathbf{v}}}{\partial t} + \frac{1}{\rho}\nabla\tilde{p} \right\rangle dt \tag{2.42}$$

Since the goal is the derivative of $\mathcal{L}$ by state variables, which are $\tilde{p}$ and $\tilde{\mathbf{v}}$, By taking the derivative of $\mathcal{L}$ by state variables equal to zero, the solution of $\boldsymbol{\lambda}$ is hope to be achieved. Hence, the arrangement of equation (2.42) is needed to make it easier. That arrangement can be done by integration by parts to swap the state variables out of temporal derivatives, and product rules for spatial derivation. Considering temporal derivative

$$\int_0^T \left\langle \tilde{\lambda}_p, \frac{\partial \tilde{p}}{\partial t} \right\rangle dt = \int_0^T \int \tilde{\lambda}_p \frac{\partial \tilde{p}}{\partial t} dxdt$$

Applying integration by parts by define $u = \tilde{\lambda}_p$ and $dv = \frac{\partial \tilde{p}}{\partial t} dt$

$$\begin{aligned} \int_0^T \int \tilde{\lambda}_p \frac{\partial \tilde{p}}{\partial t} dxdt &= \tilde{\lambda}_p \tilde{p}|_0^T - \int_0^T \int \tilde{p} \frac{\partial \tilde{\lambda}_p}{\partial t} dxdt \\ &= -\int_0^T \left\langle \tilde{p}, \frac{\partial \tilde{\lambda}_p}{\partial t} \right\rangle dt \end{aligned} \tag{2.43}$$

The first term of equation (2.43) is depleted by the initial condition of p ($p(0) = 0$) and the terminal condition $\tilde{\lambda}_p(T) = 0$. Similarly,

$$\int_0^T \left\langle \tilde{\lambda}_{\mathbf{v}}, \frac{\partial \tilde{\mathbf{v}}}{\partial t} \right\rangle dt = -\int_0^T \left\langle \tilde{\mathbf{v}}, \frac{\partial \tilde{\lambda}_{\mathbf{v}}}{\partial t} \right\rangle dt \tag{2.44}$$

For the spatial derivation, applying product rule grants

$$\int_0^T \int \tilde{\lambda}_p K \nabla \cdot \tilde{\mathbf{v}} dxdt = \int_0^T \int \nabla \cdot \left(K \tilde{\lambda}_p \tilde{\mathbf{v}} \right) dxdt - \int_0^T \int \tilde{\mathbf{v}} \cdot \nabla \left(K \tilde{\lambda}_p \right) dxdt$$

Notice that the first term can be changed to surface integral, and an approximation of the dot product between vector field and surface integral is equal to zero ($\tilde{\mathbf{v}} \cdot da = 0$) has been made. Hence, this spatial derivative term will be in the form of

$$\int_0^T \left\langle \tilde{\lambda}_p K \nabla \cdot \tilde{\mathbf{v}} \right\rangle dt = -\int_0^T \left\langle \tilde{\mathbf{v}} \cdot \nabla \left(K \tilde{\lambda}_p \right) \right\rangle dt \tag{2.45}$$

Similary,

$$\int_0^T \left\langle \tilde{\lambda}_{\mathbf{v}} \frac{1}{\rho} \nabla \tilde{p} \right\rangle dt = -\int_0^T \left\langle \tilde{p} \nabla \left(\frac{1}{\rho} \tilde{\lambda}_{\mathbf{v}} \right) \right\rangle dt \tag{2.46}$$

Put equations (2.43), (2.44), (2.45), and (2.46) back to equation (2.42) yields

$$\begin{aligned} \mathcal{L}(\tilde{u}, \tilde{\lambda}, m) = & \int_0^T \int \frac{1}{2} (D-d)^2 dx dt + \int_0^T \left\langle \tilde{p}, \frac{\partial \tilde{\lambda}_p}{\partial t} \right\rangle dt \\ & + \int_0^T \left\langle \tilde{\mathbf{v}}, \nabla \left(K \tilde{\lambda}_p \right) \right\rangle dt + \int_0^T \left\langle \tilde{\mathbf{v}}, \frac{\partial \tilde{\lambda}_{\mathbf{v}}}{\partial t} \right\rangle dt \\ & + \int_0^T \left\langle \tilde{p}, \nabla \cdot \left(\frac{1}{\rho} \tilde{\lambda}_{\mathbf{v}} \right) \right\rangle dt \end{aligned} \tag{2.47}$$

Then, take a derivative of $\mathcal{L}$ in equation (2.47) with respect to $\tilde{p}$ and $\tilde{\mathbf{v}}$ equal to zero.

$$\frac{\partial \mathcal{L}}{\partial \tilde{p}} |_{p, \lambda_p, \lambda_{\mathbf{v}}} = (D-d) + \frac{\partial \lambda_p}{\partial t} + \nabla \cdot (b \lambda_{\mathbf{v}}) = 0 \tag{2.48}$$

$$\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{v}}} |_{\mathbf{v}, \lambda_p, \lambda_{\mathbf{v}}} = \frac{\partial \lambda_{\mathbf{v}}}{\partial t} + \nabla \cdot (K \lambda_p) = 0 \tag{2.49}$$

Equations (2.48) and (2.49) are in the form of wave equation similar to first-order coupled wave equation (equations (2.37) and (2.38)). With the terminal condition $\lambda_p(T) = 0$ and $\lambda_{\mathbf{v}}(T) = 0$, these wave equations of adjoint-state variable should be solved by backward time stepping. These equations can also be shifted to make it looks exactly like wave equations by multiply equation (2.48) with K, and define another state variables which are $\lambda'_p = K \lambda_p$ and $\lambda'_{\mathbf{v}} = \frac{1}{\rho} \lambda_{\mathbf{v}}$

$$\frac{\partial \lambda'_p}{\partial t} + K \nabla \cdot \lambda'_{\mathbf{v}} = -K(D-d) \tag{2.50}$$

$$\frac{\partial \lambda'_{\mathbf{v}}}{\partial t} + \frac{1}{\rho} \nabla \lambda'_p = 0 \tag{2.51}$$

After the adjoint-state variables are identified, the gradient of functional is easily obtained by the derivative of $\mathcal{L}$ with respect to K.

$$\frac{\partial \mathcal{L}}{\partial K} = \frac{\partial h}{\partial K} - \int_0^T \left\langle \boldsymbol{\lambda}, \frac{\partial F}{\partial K} \right\rangle dt \tag{2.52}$$

With the only term that consists of K in forward modeling, the gradient of objective function will be

$$\frac{\partial \mathcal{L}}{\partial K} = -\int_0^T \Big\langle \lambda_p, \nabla \cdot \mathbf{v} \Big\rangle dt \tag{2.53}$$

Solutions from equations (2.50) and (2.51) are the receiver wavefields from the residual data which similar to the receiver wavefields from seismogram in RTM. Moreover, the gradient of objective function in equation (2.53) is a zero-laggged crosscorrelation of source and receiver wavefields. Hence, the adjoint-state method can be viewed as a mathematical approach of RTM method which resemblant to the work of Tarantola (1984). In addition, equation (2.32) states that by applying operator $\mathbf{L}^T$ to the residual data, the gradient can be achieved. Hence, operator $\mathbf{L}^T$ acts like RTM algorithm and equation (2.32) is applying RTM method to the residual data. Similarly, RTM method in matrix form can be written as $\mathbf{m} = \mathbf{L}^T\mathbf{D}$.

Every principles and methods that explained in this chapter can be concluded into the flowchart of the LSRTM method in Figure 2.4. Start off with the image **m** that achieved from them migration or the transpose of forward modeling operator **L** of observed data **D**. Then, simulated data **d** will be generated by the migrated image modeling of the RTM image. This data will be compared to the observed data to make a residual data. By doing a migration to the residual data, the gradient of objective function **g** can be achieved. The gradient will be adjusted for better convergence rate by conjugate gradient method into an adjusted gradient **z**. The model will be updated by **z** and step length α, and the loop is going on until the maximum number of iteration is met.

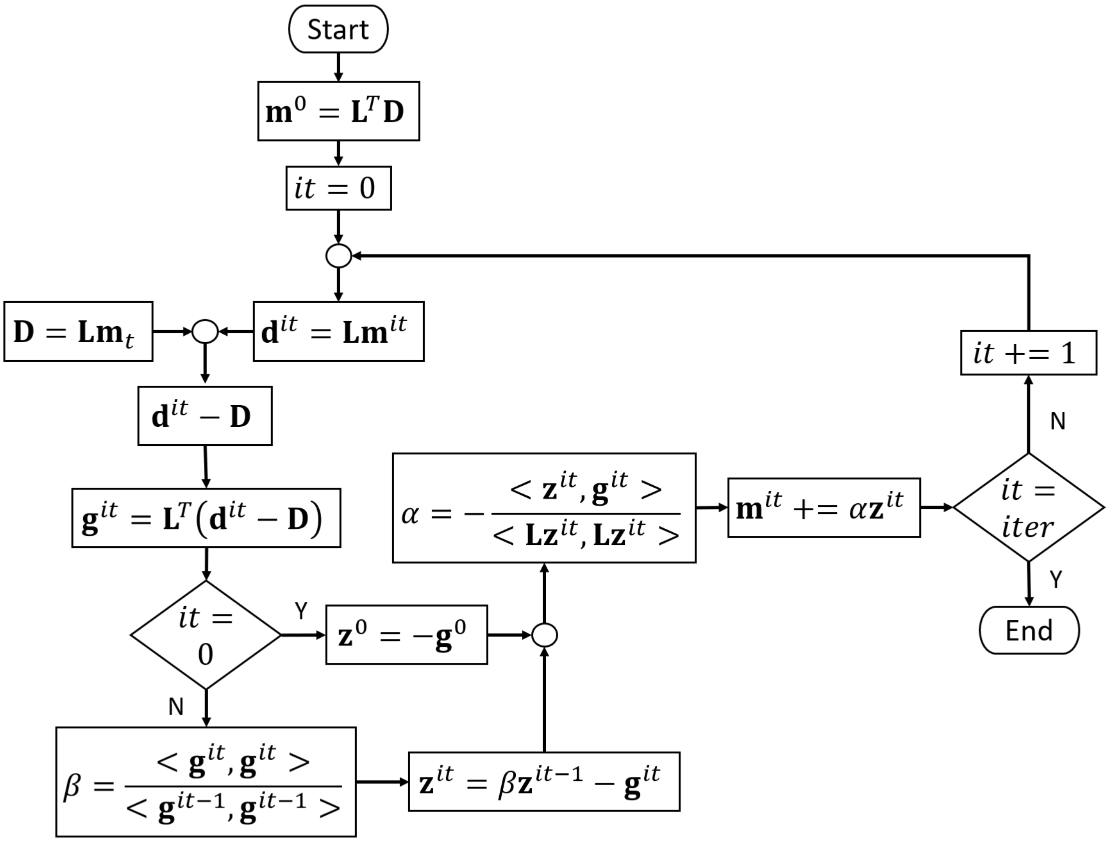

Start
$\mathbf{m}^0 = \mathbf{L}^T\mathbf{D}$
$it = 0$
$\mathbf{D} = \mathbf{L}\mathbf{m}_t$
$\mathbf{d}^{it} = \mathbf{L}\mathbf{m}^{it}$
$\mathbf{d}^{it} - \mathbf{D}$
$\mathbf{g}^{it} = \mathbf{L}^T(\mathbf{d}^{it} - \mathbf{D})$
$it = 0$
Y
N
$\mathbf{z}^0 = -\mathbf{g}^0$
$\beta = \frac{<\mathbf{g}^{it}, \mathbf{g}^{it}>}{<\mathbf{g}^{it-1}, \mathbf{g}^{it-1}>}$
$\mathbf{z}^{it} = \beta\mathbf{z}^{it-1} - \mathbf{g}^{it}$
$\alpha = -\frac{<\mathbf{z}^{it}, \mathbf{g}^{it}>}{<\mathbf{L}\mathbf{z}^{it}, \mathbf{L}\mathbf{z}^{it}>}$
$\mathbf{m}^{it} \mathrel{+}= \alpha\mathbf{z}^{it}$
$it = iter$
N
Y
$it \mathrel{+}= 1$
End

Figure 2.4: Flowchart of LSRTM method, where it is iteration index.

CHAPTER III
IMPLEMENTATION

Every methods in Chapter 2 are implemented in the GPU-based LSRTM method source code name PS_LSRTM. The further detail of the implementation is going to be explained in this chapter. Starting off with the GPU implementation, then each part of PS_LSRTM algorithm will be clarified.

3.1 GPU Implementation

Comparing CPU and GPU architecture in Figure 3.1 shows that GPU resources are devoted to data processing more than CPU. Therefore, the algorithm which deals with a large pile of data like LSRTM should perform better using GPU programming. In addition, the efficiency of GPU programming can be depicted in Figure 3.2.

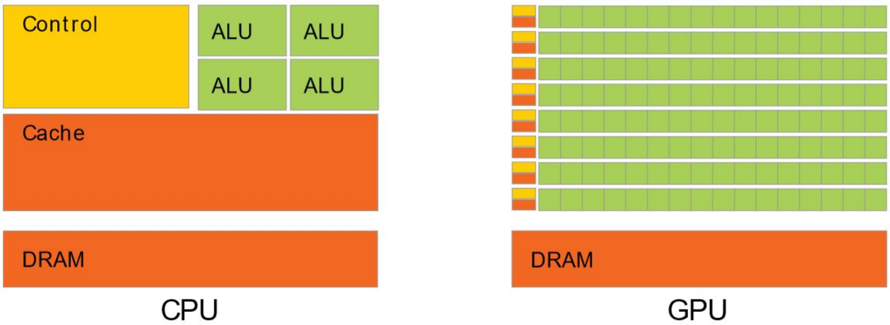


Figure 3.1: CPU and GPU architecture (NVIDIA, 2018)

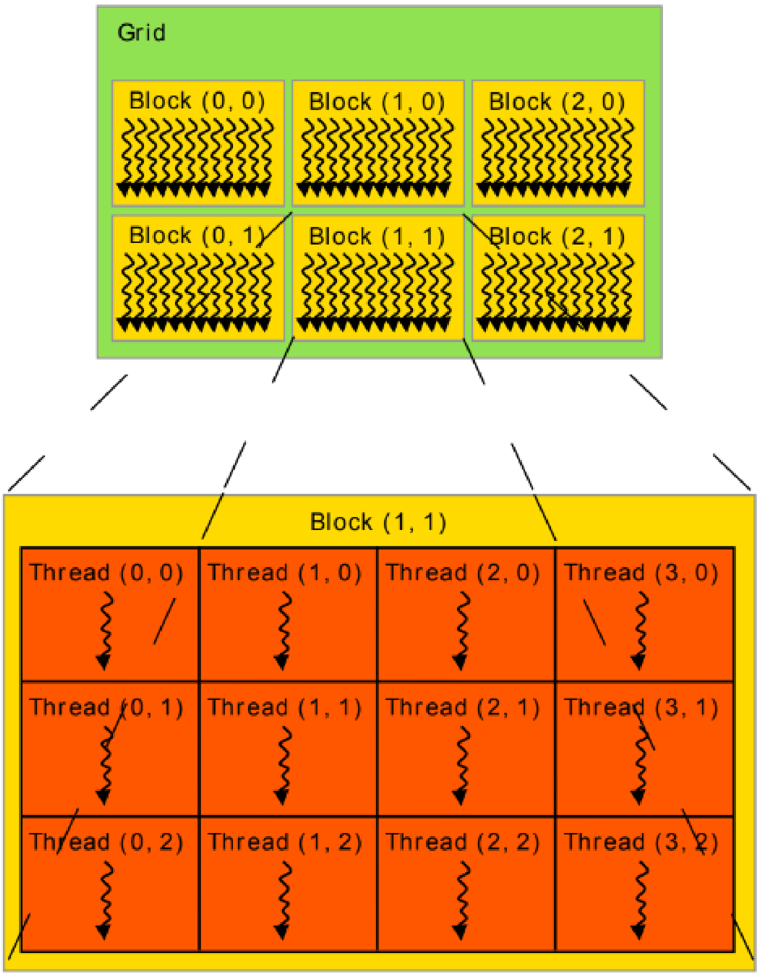
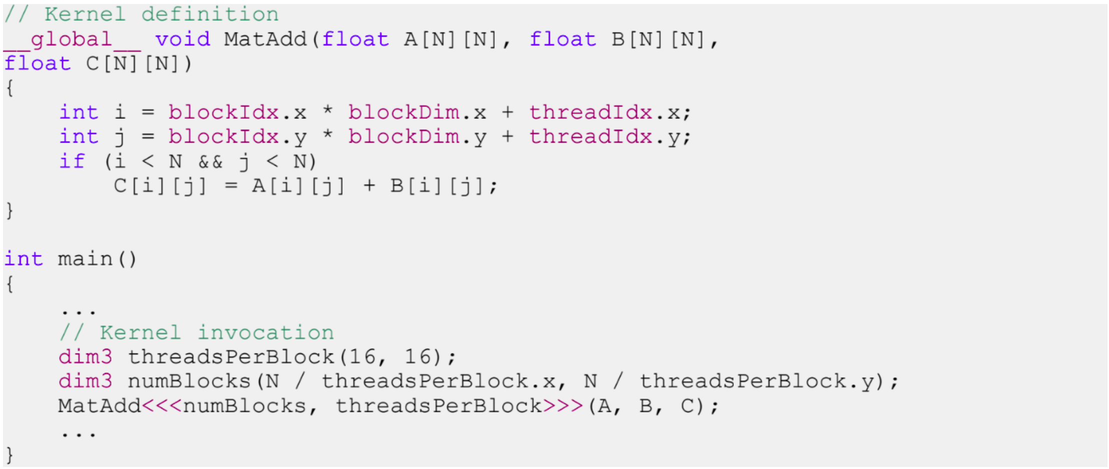


Figure 3.2: Grid of blocks and block of threads (NVIDIA, 2018)

A black arrow is a thread which represents a process in programming. For serial programming, one thread is executed at a time. On the other hand, thousands of threads can be executed concurrently which should result in a better performance. By using the NVIDIA CUDA programming model, there are blocks and grids that partitioning the threads and blocks as seen in Figure 3.2, respectively. The block can be named as block of threads, while the grid can be called grid of blocks. The block has its own memory resource that can be accessed between blocks which is shared memory. While, the grid has a larger memory resource called global memory which takes a longer time to access. Each thread in a block and a grid can be identified and individually accessed through the CUDA built-in variable $threadIdx$, $blockIdx$, and $blockDim$. By doing so, a serial code can be easily transformed into a parallel code. A prime example is presented with a code of two-demensional matrix additional in Figure 3.3. Noticing that instead of two for loops, indices i and j are indexed by CUDA built-in variables. With all indices are accessed, adding two matrices together becomes a one-line command. Figure 3.3 also shows another main part of GPU implementation by CUDA which is kernel function. Kernel function can be viewed as a C-language function with the execution configuration syntax $<<<$ ... $>>>$. This syntax is used for specification of the number of block (B) and grid (G). The set of threads that will be processed in the kernel will be divided according to G and B which can be either $dim3$ or int variables.



```
// Kernel definition
__global__ void MatAdd(float A[N][N], float B[N][N],
float C[N][N])
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    if (i < N && j < N)
        C[i][j] = A[i][j] + B[i][j];
}

int main()
{
    ...
    // Kernel invocation
    dim3 threadsPerBlock(16, 16);
    dim3 numBlocks(N / threadsPerBlock.x, N / threadsPerBlock.y);
    MatAdd<<<numBlocks, threadsPerBlock>>>(A, B, C);
    ...
}
```

Figure 3.3: An example code of adding matrices in CUDA (NVIDIA, 2018)

As informed that CUDA programming uses kernel to shift the process execution onto GPU, it is also meant that the other part of program is still run on CPU. The program that consists of serial code that run on host (CPU) and parallel code that run on device (GPU) can be called as heterogeneous programming which is depicted by Figure 3.4. With this structure, programming in CUDA needs both host and device memory allocation seperately. In consequences, a transfer of data between host and device is needed and it can be done by using intrinsic function $cudaMemcpy$. Another trait of CUDA programming model is also show in Figure 3.4, that trait is the scalable programming. The scalable programming model allows the device with more multiprocessor to execute the program with faster computational time than that of the less one (NVIDIA, 2018).

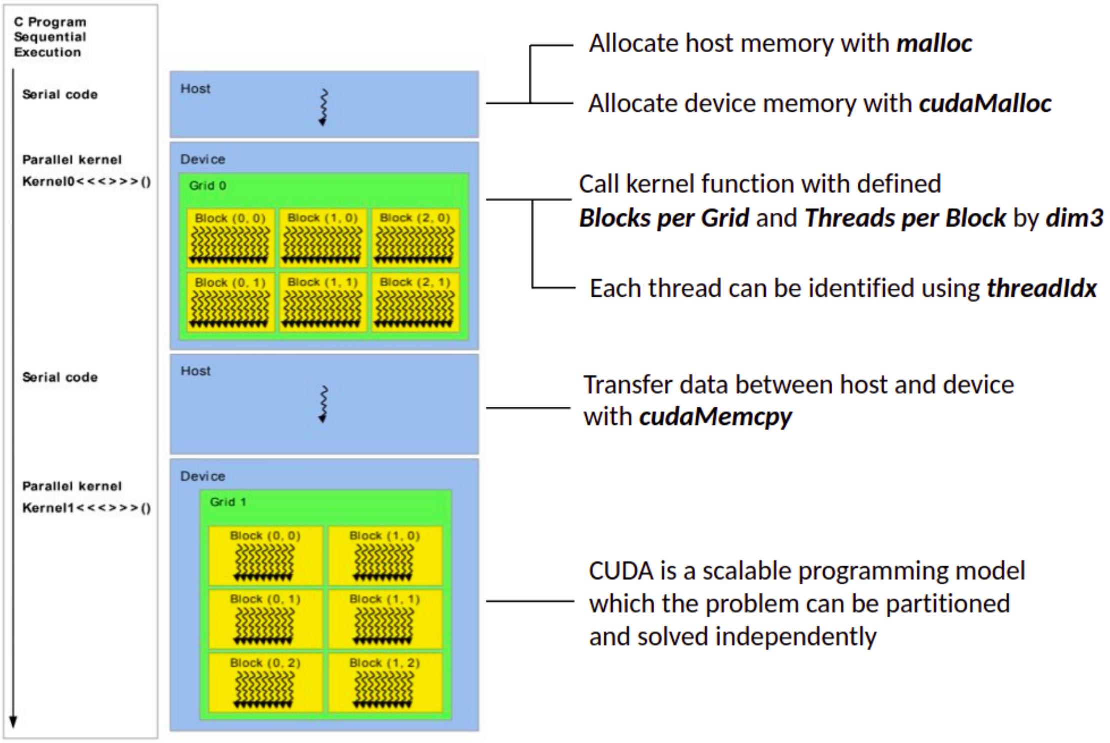


Figure 3.4: Sequential execution of heterogenous programming with the detail of CUDA built-in variables and function usage (NVIDIA, 2018).

Following Micikevicius (2009), the major part of GPU implementation will be applied on FD calculation. In FD calculation, nodes are called for a process redundantly. In Figure 3.5, 24 neighbor nodes (halos) has to be accessed for calculating a red target node for a 8^{th}-order three-dimensional FD stencil. Moreover, the communication between host and device memory heavily affects the computational runtime. Therefore, this problem is needed to be dealth with, and the chosen solution is the usage of shared memory.

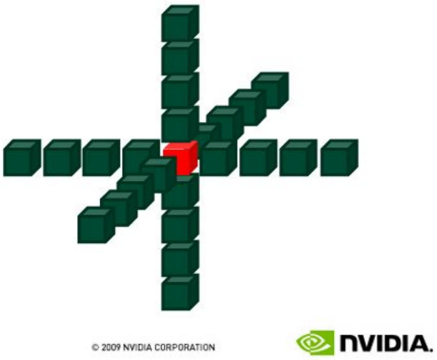


Figure 3.5: 8^{th}-order 3D FD Stencil, where red cube represents the target node (NVIDIA, 2018)

Unlike the work of Micikevicius (2009), the governing equations in this work are the first-order coupled acoustic wave equations that solved by the second-order FD on staggered gird. Therefore, the form of data tiles like Figure 1.5 must be changed. According to the FD stencil in Figure 2.2, the data tile in the shared memory that used in the FD calculation is depicted by Figure 3.6, where u and w represent v_z and v_x, respectively. B in the blue box is the block size that defined in the kernel, while the halo boxes size are a node each.

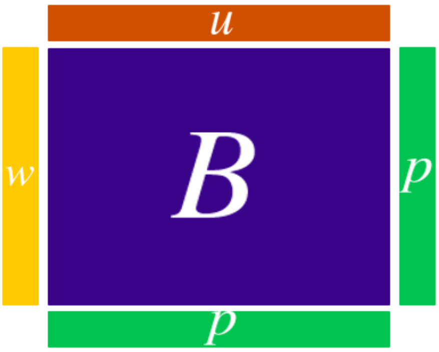


Figure 3.6: Shared memory used in FD stencil for staggered grid of the first-order coupled acoustic wave equation

To maximize the shared memory capability, an ideal implementation is to define the block size to the maximum threads per block. However, that naive method cannot be done due to the halos. Using the device used in this work as an example, if the maximum threads per block is 1024, assign the size of B as 32×32 will result into (32+2)×(32+2)-

4 which is 1124 threads. Moreover, the appropriate size should be dividable by the number of warp size which is 32 in this device. The warp size tells how many threads are accesed for processing at the same time. With these conditions, the block size for FD kernel in this work will be set as 24×24.

3.2 PS_LSRTM Algorithm

A GPU-implemented LSRTM algorithm, PS_LSRTM, is constructed for the purpose of testing the LSRTM capability in imaging and also testing the speedup of computation runtime of a GPU-based code to a CPU-based code. A workflow of PS_LSRTM is presented in Figure 3.7, this workflow is used to operate the LSRTM algorithm in Figure 2.4. Next, each procedure in PS_LSRTM algorithm will be explained seperately in each section in this chapter.

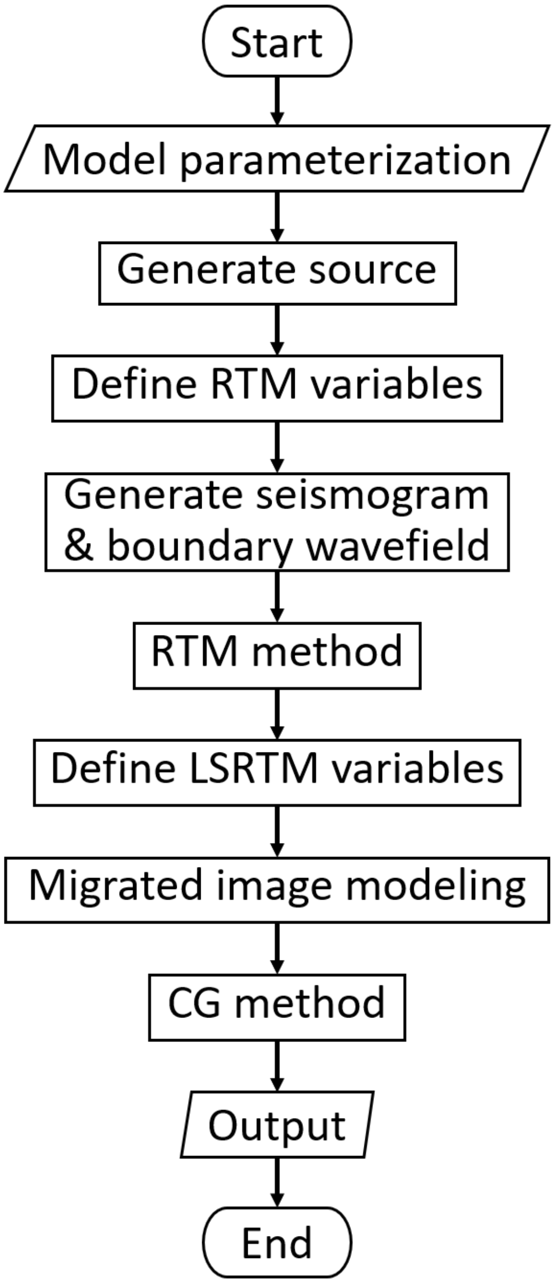


Figure 3.7: PS_LSRTM workflow

3.3 Model Parameterization

Model parameterizarion is a process that define the required model parameters for the algorithm. First, a velocity model will be defined or loaded into the program first. The variables that needed to be defined to cope with the model are model size horizontally (nx), vertically (nz), and mesh size (h). Apart from those variables, there is another one that need to be defined which is boundary length (b) for PML boundaries. The model will be padded at every sides by b grids with the same velocity as the boarder grid and damping function ($\sigma(\mathbf{x})$). Therefore, the extended model will have a size of nnx and nnz which calculated from $nx+2(b+1)$ and $nz+2(b+1)$. The P-wave velocity at every point of the model (c) will be used to calculate the bulk modulus K with the assumption of constant density (ρ) by a formular $K = c^2/\rho$. After the parameterizarion has done, the model will be smoothed to be used as an initial model. In this study, a selected method for smoothing the model is a Gaussian blur. An idea of Gaussian blur or Gaussian smoothing is a convolution of Guassian function with the image to reduce its detail and noise. The Guassian function (G), used in this work, is a two-dimensional function which has the form of

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1}$$

Note that σ here refers to standard deviation of the Gaussian distribution, and a visualization of G is shown in Figure 3.8. After the convolution the smoothed velocity model is ready to be used in furthur procedure.

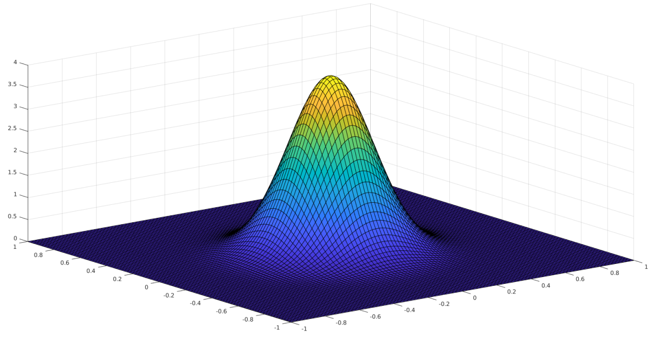


Figure 3.8: 2D Guassian distribution ($\sigma = 0.2$)

3.4 Generate Source

A duty of this process is generating source wavelet, time series, and defining a source position. The wavelet used in this work is a Ricker wavelet which is generated in a time series with its peak at $1/f$ seconds where f is an assigned dominant source frequency. Figure 3.9 shows the generated ricker wavelet with $f = 5\ Hz$.

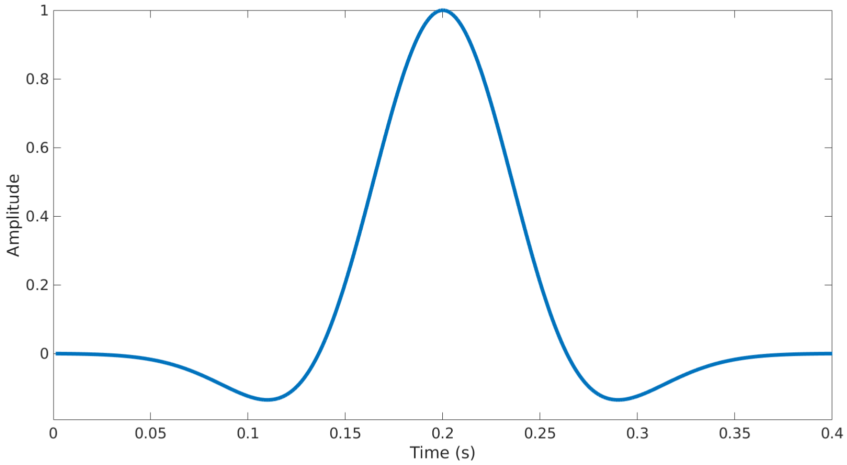


Figure 3.9: Ricker wavelet ($f = 5\ Hz$)

A time series is defined by two variables which are record length (T) and time interval (Δt). The record length is defined by an assumption of the slowest wave propagating to the bottom of the model and back to the surface. Hence, T is calculated by a two times model depth (d) divided by the least value of c ($T = 2d/c_{min}$). dt is calculated based on numerical stability according to Virieux (1986) which is $h/\sqrt{2}c_{max}$, but the input Δt can be scaled down by any particular factor. After T and Δt are defined, the time series can be produced by stepping up time value by Δt from 0 to T. To define source position in two-dimensional model, a coordinate of horizontal and vertical point is needed. In this source code, the vertical point is fixed at the surface ($sz = 0$), and the horizontal points (sx) are evenly-spread troughout the surface.

3.5 RTM Variables

As its name states, this section is created to define necessary variables for the RTM method such as sources wavefield, receivers wavefield, seismogram, boundaries wavefield, and RTM image. There are also variables that needed in CUDA API which are block and grid size. The block size (B) used in this are 24×24, and the grid size are defined by the problem size divided by 24. There are two grid size variables which are G and Gr. G is accounted for the problem size $nnx \times nnz$, while Gr is accounted for the problem size $nx \times nz$. Since this is a heterogeneous programming, these variables are needed to be stored in both host and device memory. The host memory allocation can be done by using function $malloc$ for C programming, while the device memory allocation can be done by using function $cudaMalloc$. However, $struct$ variables can not be allocate directly using $cudaMalloc$. This problem can be solved by allocating the host variables by $cudaMalloc$ and copy those memories by using $cudaMemcpy$ to the device variables. $struct$ variables used in this code are seismogram ($seis$) from real velocity model and boundaries wavefield (pn). The array of $seis$ is used to segregate the seismogram from sources in different position, while pn is used to store pressure wavefield in each side of the boundaries (top, bottom, right, and left). This boundaries wavefield $struct$ type was also applied to the others wavefield as well.

3.6 Generate Seismogram

The seismogram can be generated by the forward modeling scheme that simulates the propagation of source wavefield throughout the model. The seismogram generated in this step is used as an observed shot record data, so it should be acquired from the forward modeling of the true velocity model. A flowchart of this procedure is shown in Figure 3.10. An initial wavefield will be initialized first before entering the time loop. The advance timestep procedure refers to the FD calculation that results in the value of the variable at the next timestep which can be seen in equation 3.2 as an example.

$$p^{k+1}_{x_{i,j}} = p^{k}_{x_{i,j}} - \Delta t \sigma_{i,j} p^{k}_{x_{i,j}} - \frac{\Delta t K_{i,j}}{\Delta x}\left(v^{k+1/2}_{x_{i+1/2,j}} - v^{k+1/2}_{x_{i-1/2,j}}\right) \tag{3.2}$$

This FD calculation is done the kernel with the problem size of B and G as stated earlier. After the calculation is done, the next step is applying the source amplitude to the wavefield. The source injection is also called by a kernel, but it is called by only a thread to perform an action to increase the amplitude of p wavefield at source point by an amplitude of source wavelet at time step k. Then, the pressure wavefield at every receivers location will be stored to make a shot record data. After that, the initial value of every wavefield will be replaced by the calculated data with the kernel with the block and grid size B and G. At the end of time and source loop, the stack seimogram, which is a pile of seismogram from every source, will be exported for an output file.

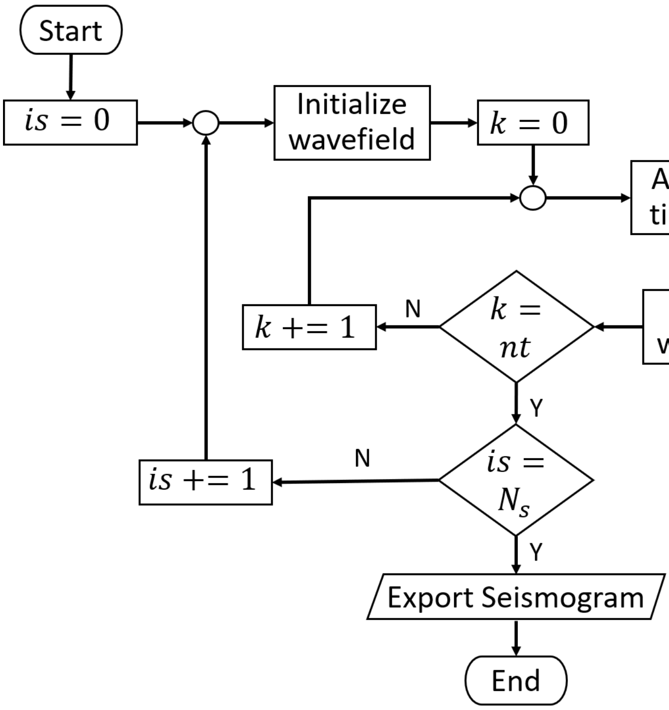


Figure 3.10: A flowchart of the forward modeling scheme

3.7 RTM Method

The RTM algorithm starts with the forward modeling of smooth model for saving the boundary wavefield at every time step which has the workflow like that of Figure 3.10. Then, the backward time loop is started for the RTM algorithm. The boundary wavefield is imported at every time step for the reconstruction of source wavefield, and the seismogram is also imported as a source for backward-propagation receiver wavefield. The boundary wavefield already contain the source amplitude, so it is depleted in the eject source step. Instead of doing the FD calculation for increase time index, the FD calculation is done for the decrease time index in the backward timestep procedure. The example of the FD calculation of reconstructed wavefield is shown at equation 3.3.

$$p_{x_{i,j}}^{k-1} = p_{x_{i,j}}^{k} - \Delta t \sigma_{i,j} p_{x_{i,j}}^{k} - \frac{\Delta t K_{i,j}}{\Delta x} \left(v_{x_{i+1/2,j}}^{k-1/2} - v_{x_{i-1/2,j}}^{k-1/2} \right) \tag{3.3}$$

Since, the backward-propagation equations are adjusted to be the form like wave equations. The FD calculation for receivers wavefield is also in the form of equation 3.3 as well. The differences are the change of variable from source to receivers wavefield, and the source is the seismogram. These calculation are also called by the kernel with problem size B and G. The reconstructed source wavefield and the backward-propation receiver wavefield are crosscorrelated for the migrated image with the kernel that has the grid and block size of Gr and B. Both source and receiver wavefield will be updated at the end of each time loop. The RTM image will be stacked at every source loop and export to host memory at the last iteration which marks the end of RTM method.

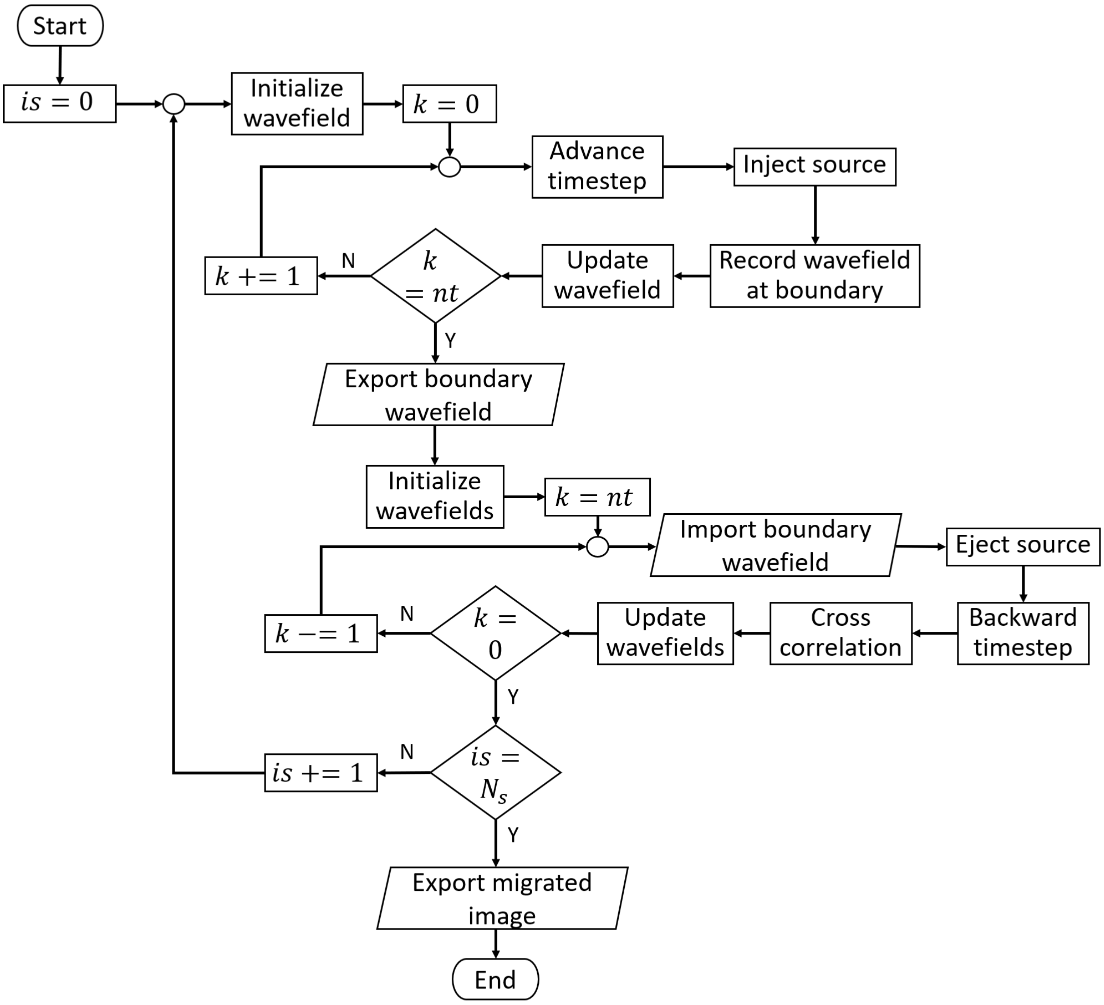


Figure 3.11: A flowchart of the RTM method

3.8 LSRTM Variables

As usual, the accounted variables and parameter will be defined and declared at the beginning of the algorithm. For LSRTM method, the perturbed wavefield variables are needed for migrated image modeling, and three more image variables for gradient of objective function, conjugate gradient, and optimized image. The gradient of objective function needs to have two parameters because there is a calculation that needs its value at two different iteration. This reason should have be the same with the conjugate gradient, but it can be technically neglected by the programming implementation. The optimized image value will be copied from RTM image first, and it will be optimized by the CG algorithm later. There are also variables that used in optimization such as two-norm squares of residual data that indicates the convergence rate, alpha (α), and beta (β). The calculation of residual data and α require the seismogram from migrated image modeling, so another seismogram variable is also defined. The only parameter in this step is an iteration number ($iter$) that limit the optimization loop.

3.9 Migrated Image Modeling

Apart from simulating the perturbed wavefield, this step also need to simulating the source wavefields for the secondary source used in modeling from RTM image. Hence, the migrated image modeling flowchart, shown in Figure 3.12 displays two advance time stepping schemes. The first one is for the source wavefields. Then, the wavefields will be injected with source and recorded on boundaries. Next, the second one is called for modeling image. As same as the previous FD calculation, the kernels used here are called with the size of B and G. The perturbed wavefields will be added by the multiplication of migrated image, bulk modulus, and a temporal derivative of velocity wavefield in the inject secondary source step. The injection of secondary source is called by the kernel size B and Gr due to the included migrated image. The perturbed wavefield is recorded for the simulated data, used in objective fucntion. After source loop is completed, the two-norm squares of residual data is calculated, and this value at each iteration will be exported to be plotted as a convergent rate graph.

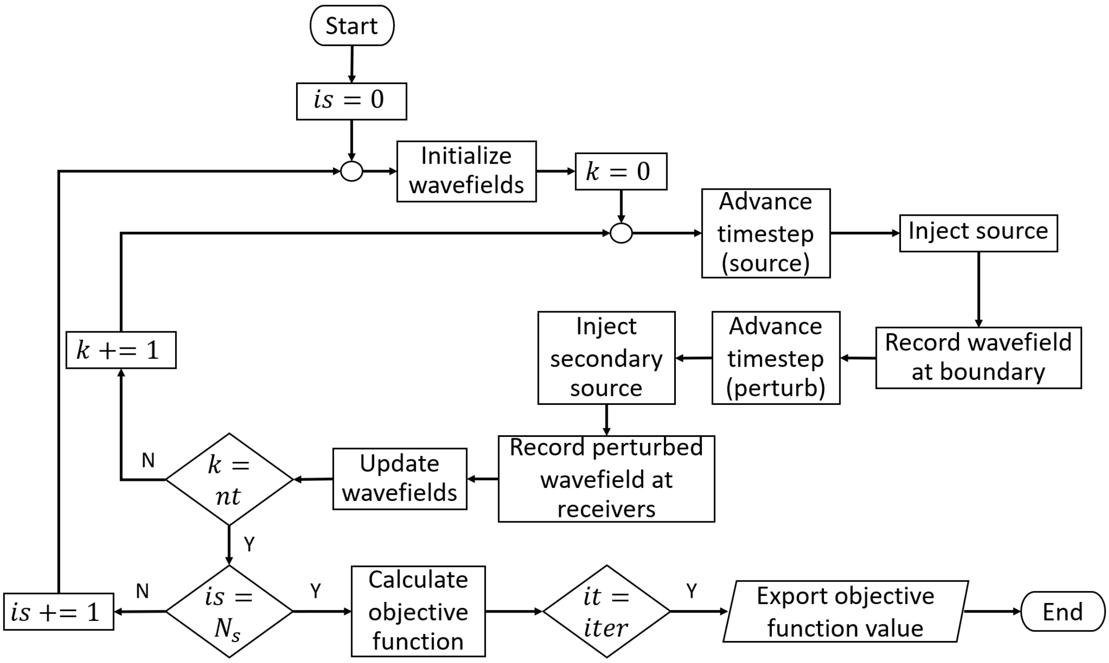

Start
$is = 0$
Initialize wavefields
$k = 0$
Advance timestep (source)
Inject source
Record wavefield at boundary
Advance timestep (perturb)
Inject secondary source
Record perturbed wavefield at receivers
Update wavefields
$k = nt$
N
Y
$k += 1$
$is = N_s$
N
$is += 1$
Y
Calculate objective function
$it = iter$
Y
Export objective function value
End

Figure 3.12: The migrated image modeling flowchart

3.10 Gradient of Objective Function Computation

It has already stated that the gradient of objective function can be achieved by applying RTM method to a residual data. Therefore, the flowchart in Figure 3.13 shows the resemblace to the flowchart of RTM method. The residual data is imported instead of the seismogram as a source of receivers wavefield. After the FD calculation and imaging condition scheme, the acquired image is the gradient of objective function.

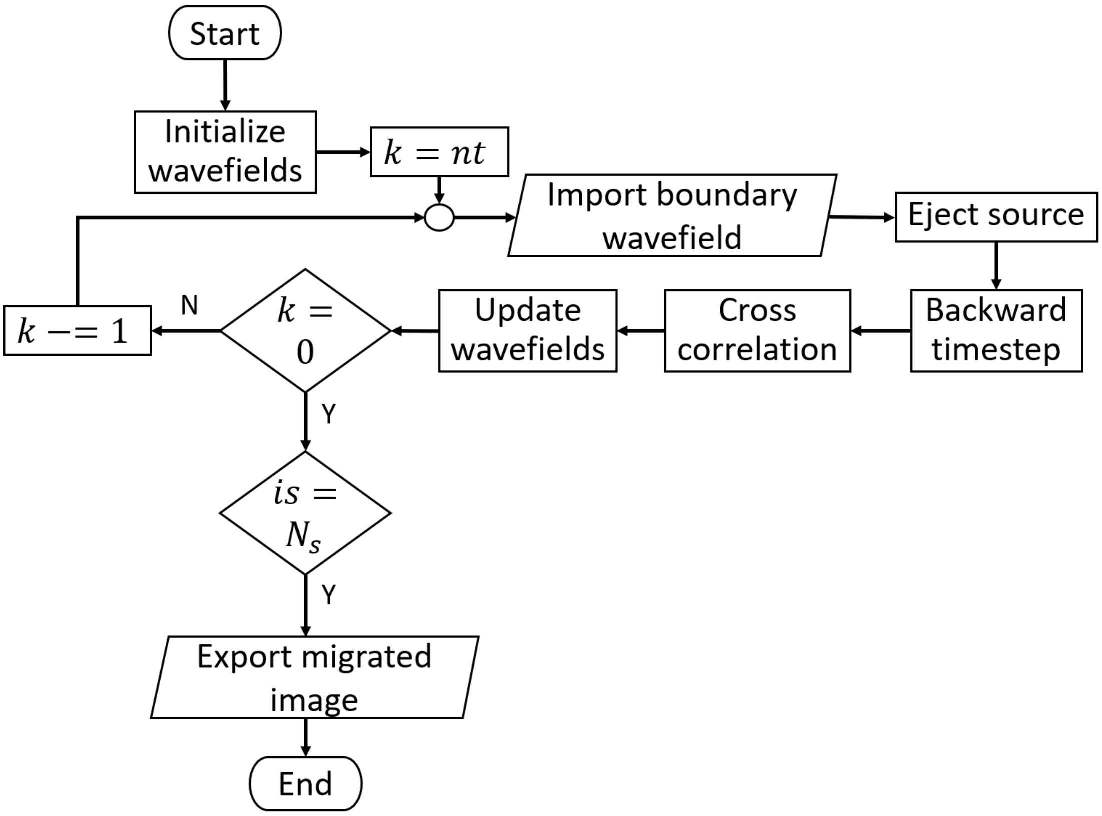


Figure 3.13: The gradient of objective function computation flowchart

3.11 Conjugate Gradient Method

The obtained gradient of objective function in the last step will go through the conjugate gradient scheme. On the first iteration of optimization loop, the conjugate gradient value will be equal to the minus value of that of the gradient of objective function. For the remaining iteration, the gradient of objective function from current and previous iteration will be used to calculate the parameter β by equation (2.33). Then, this parameter will be used to create the conjugated image by multiplying β with the gradient of objective function and updated to conjugated image according to equation (2.34). Then, the last step in an optimization iteration which is the model optimization. However, there will be another modeling algorithm before updating the model. The necessity of modeling is a denominator in the stepping parameter α. As shown in equation (2.35), the image from conjugate gradient scheme will be put through the forward modeling operation. After the modelling is completed, the parameter α will be calculated and used to optimize the model by equation (2.36). The process of inversion will continue untill it reaches the defined maximum number of iteration. Lastly, the optimized image will be exported as a result of least-squares reverse time migration.

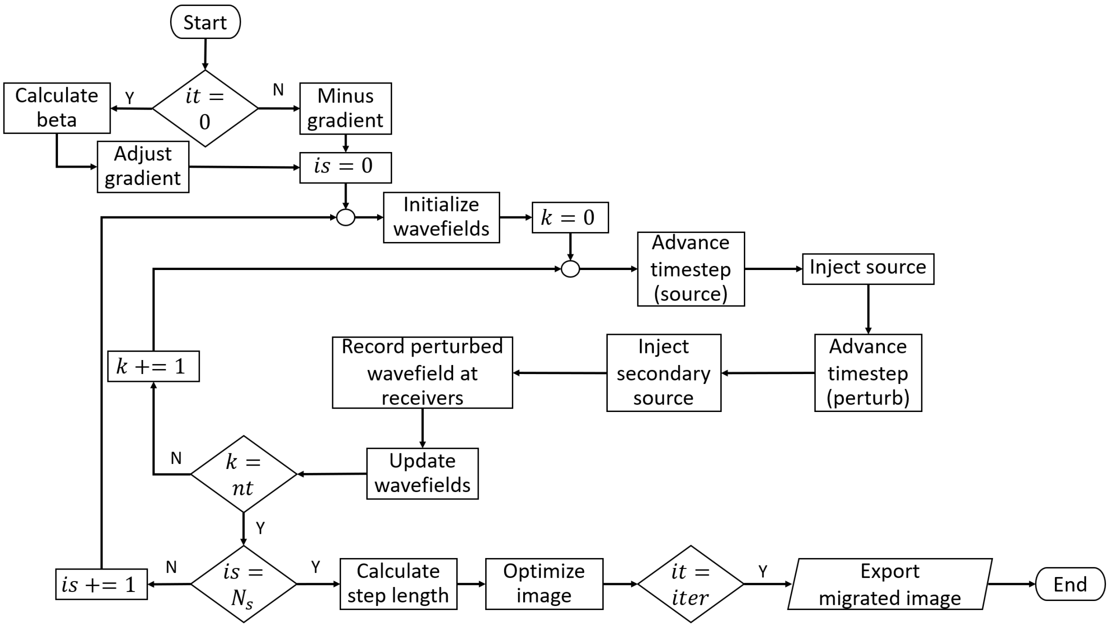


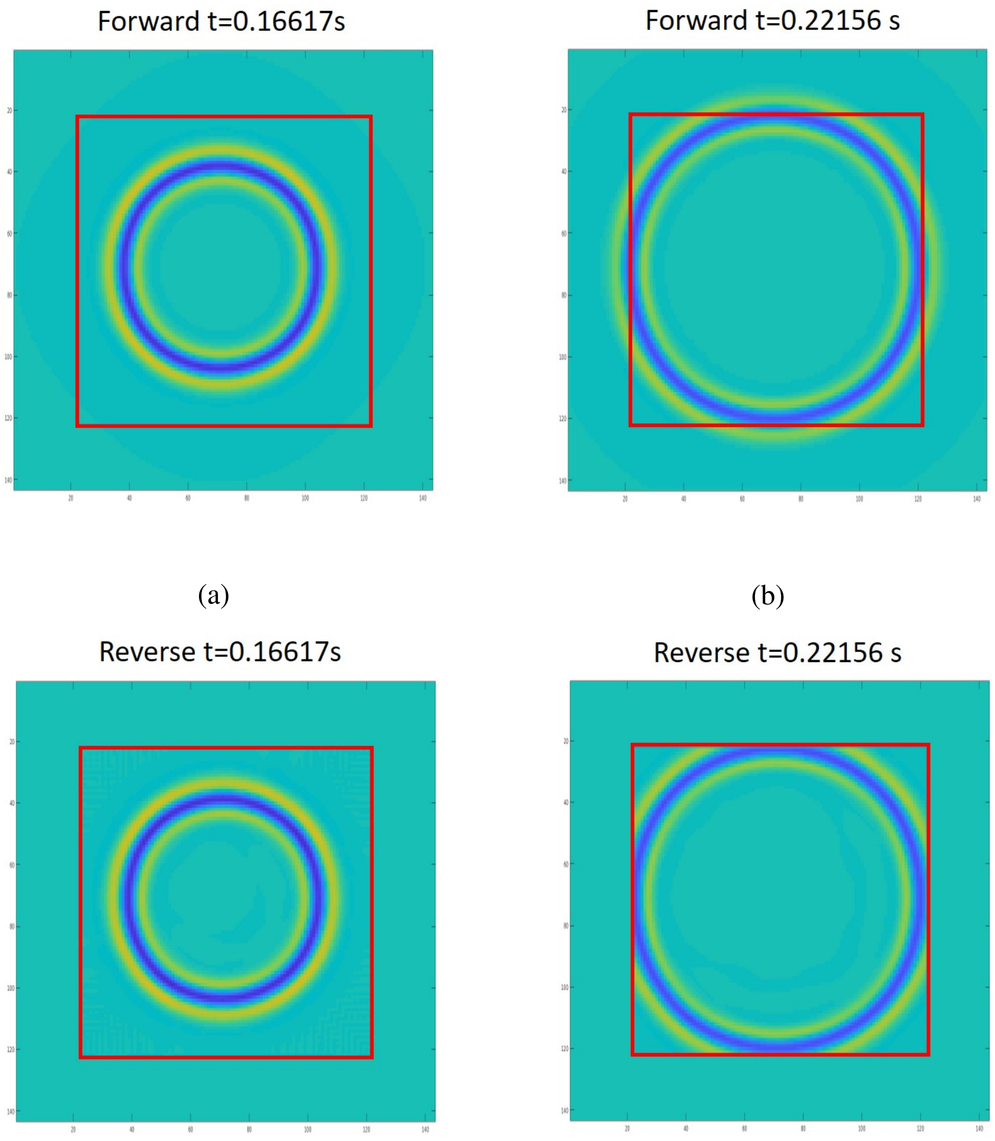
Figure 3.14: Conjugate gradient method flowchart

CHAPTER IV
NUMERICAL RESULTS

Apart from LSRTM image, there are also another results that need to be considered. The first one is a wavefield simulation in a homogeneous medium for wavefield reconstruction test. Following by the seismogram which is the result of forward modeling. Then, there are the migrated images that include migrated image (RTM image) and optimized migrated image (LSRTM image). Comparing RTM and LSRTM image can be visualized by convergence plot which shows the squares of two-norm of residual data at each iteration. Lastly, the speedup ratio acquired from the GPU implementation is shown.

4.1 Wavefield Reconstruction

By simulating wavefield propagation in the homogeneous medium, the form of propagated wave and also the effectiveness of PML boundaries can be observed as seen in Figure 4.1a and 4.1b. In Figure 4.1a, a circular wavefield is displayed as expected from a propagating wavefield from a point source. In Figure 4.1b, the wavefield starts to hit the borders which is the red square. While entering PML region, the effect of PML absorption can be seen by a faded yellow band of wavefield. Next, the wavefield will be simulated inward from the boundaries. An expected result would be an identical waveform of both outward and inward wavefield which is exactly what Figure 4.1 shows.



(a) (b)

(c) (d)

Figure 4.1: Wavefield propagation in a homogeneous medium (a) shows a forward propagation at 0.16617 s and (b) at 0.22156 s. A backward propagation at 0.22156 s and 0.16617 s are displayed at (d) and (c), respectively. A red square line indicate the physical region.

4.2 Two-Horizontal Layered Model

After the wavefield is verified, LSRTM algorithm will be tested on a simple model which is a two-horizontal layered model. A strong point of simple model is its results can be effortlessly analyzed. In this section, seismogram, migration image, and optimized migration image will be finely break down. Our two-horizontal layered model will be shown in Figure 4.2 where a true velocity model is on the left panel and the smooth version is on the right panel. The first layer has P-wave velocity value of 600 m/s while that of the lower one is 1200 m/s. The discontinuity in the true velocity model will be smoothed by Guassian filter and becomes a gradually increase velocity value region in the smooth velocity model. The model grid mesh is 101×101 with the grid size of 2 m, PML boundary consists of 20 grids.

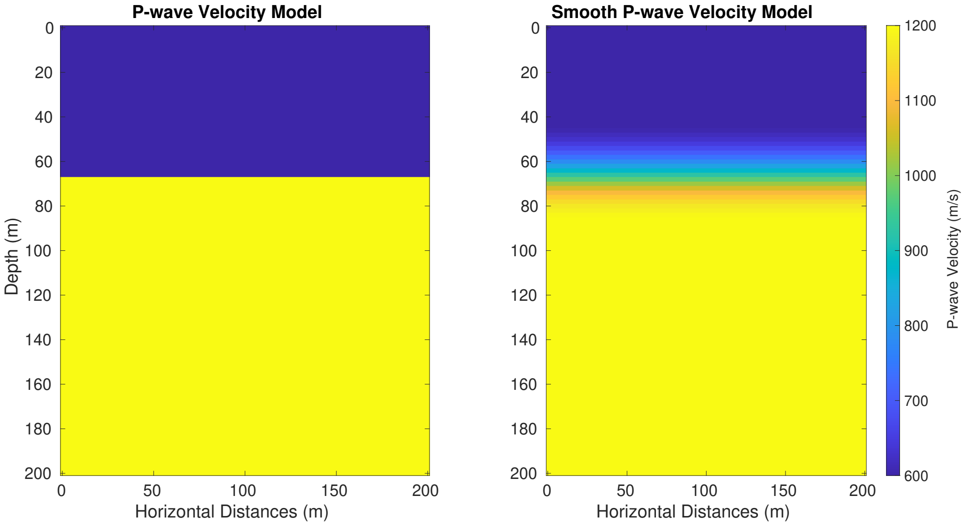


Figure 4.2: P-wave velocity model (left) and smooth P-wave velocity model (right) of the two-horizontal layered model

Next, forward modeling scheme was operated on true velocity model with a 20-Hz source at position $nx = 50$ and $nz = 0$ which is on the middle of the surface. A seismogram produced from this operation is shown in Figure 4.3. There are three seismic events detected in this seismogram which are direct wave, reflected wave, and boundary reflection wave, which labeled as D and R in Figure 4.3. Direct wave is a wave that propagates directly from the sources to the receivers. Therefore, Direct wave will travel with the velocity of the first layer which results in the linear line. Reflected wave is wave that reflected from the discontinuity in the medium which is displayed as a parabola curve in the seismogram.

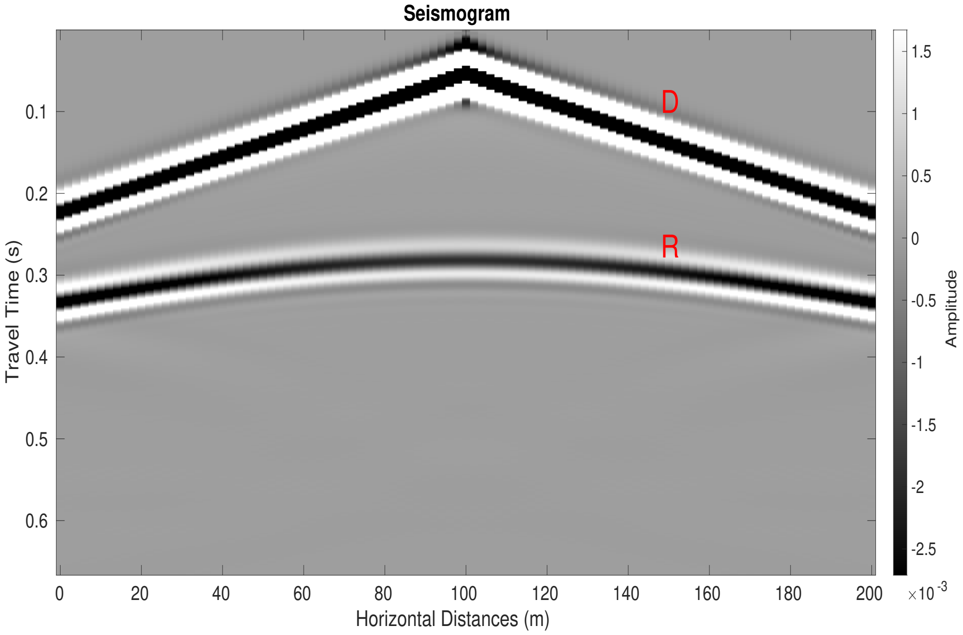


Figure 4.3: Seismogram of two-layered model with label of seismic events, D for direct wave, R for reflect wave

Then, the remaining algorithm of LSRTM were operated on the model until it was completed. The products of this operation are shown in Figure 4.4 and 4.5 which are migration image, optimized migration image, and convergence graph. The migration image on the left panel of Figure 4.4 shows two-high amplitude region which are on the surface and around 50-75 meters along the vertical direction. There is also a V-shape image between those two regions. After fifteen iterations of optimization, the optimized migratiom image on the right panel of Figure 4.4 displays only the high ampltiude region which is the one at 50-75 meters along the vertical direction. It is vivid that conjugate gradient algorithm was trying to eliminate the top surface part of the image. Since the initial model is known, the expected result is the image at the 50-75 meters along the vertical direction which both migration and optimized migration have it. However, the migration image contains the noise on the top that was later dealt with by the optimization. In addition, the success of CG algorithm is also presented by a convergence graph in Figure 4.5 which shows the exponentialy decrease of residual value in each iteration. For clarification, this characteristic depicts a decrease of residual value by CG algorithm along the opposite gradient direction. When the residual value reached its minimum, it did not rise up again even the continuing CG algorithm.

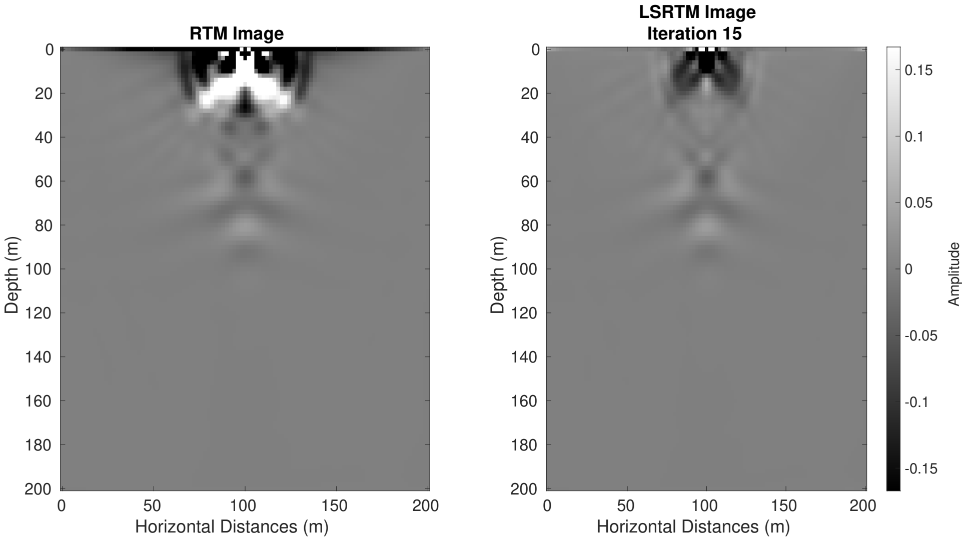


Figure 4.4: Migrated images from the LSRTM method on two-layered model with 1 20-Hz source

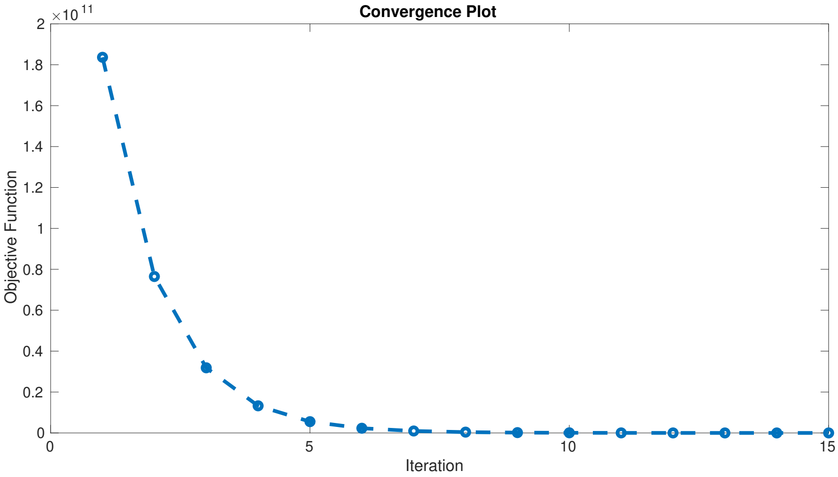


Figure 4.5: Residual graph of LSRTM on two-layered model with 1 20-Hz source

Next, the LSRTM algorithm was operated with ten 20-Hz sources distributed on the surface. The migration image on the left panel of Figure 4.6 shows the high-amplitude region on the surface with the faint straight line lies around 60 meters along the vertical direction. Despite the reduction of noise between surface and reflector from stacking, the noise on the surface is too strong that it stands out more than the reflector which is the expected image. On the other hand, the optimized image on the right panel of Figure 4.6 displays a diluted noise on the surface and the more concentrated image below. However, the continuity line is not the only part that its amplitude got boost up. Another noticable image may came from the crosscorrelation between source and receiver wavefields at the area that is not the reflector. Inspite of this problem, the residual graph in Figure 4.7 still shows the convergence.

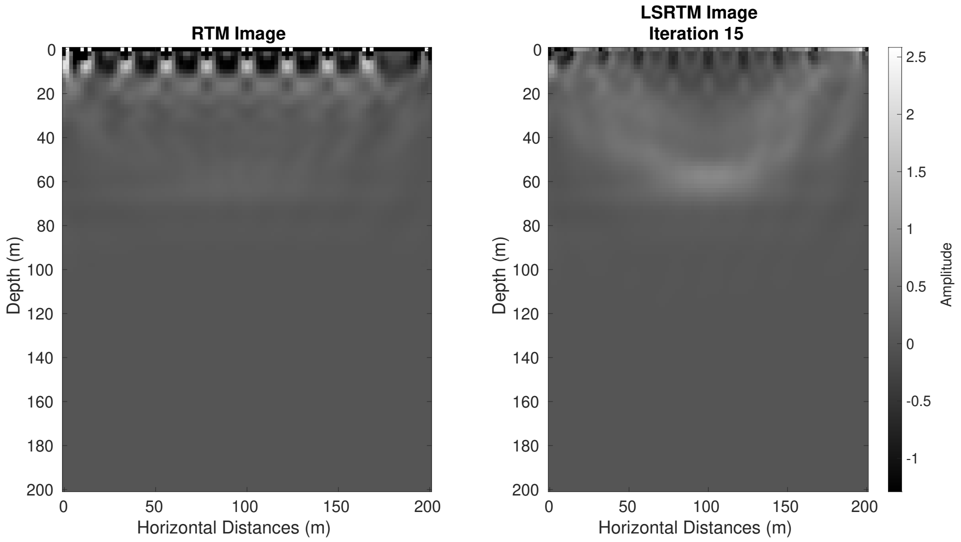


Figure 4.6: Migrated images from the LSRTM method on two-layered model with 10 20-Hz sources

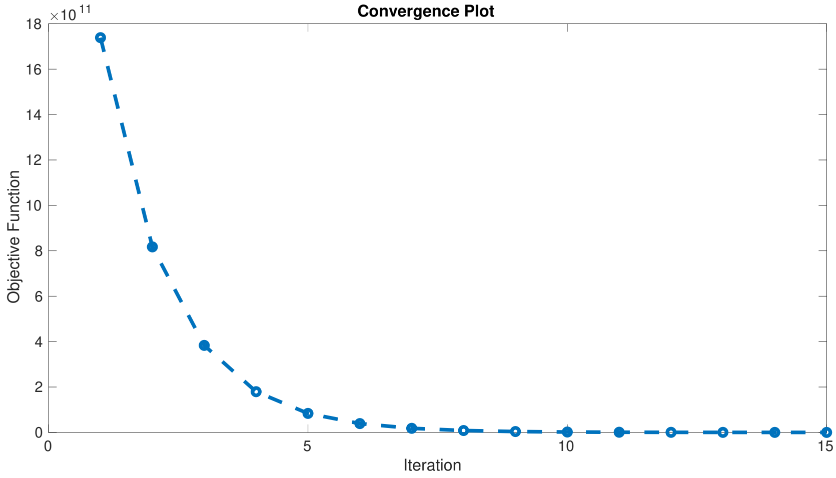


Figure 4.7: Residual graph of LSRTM on two-layered model with 10 20-Hz sources

Since RTM and LSRTM images are both generated from operator $\mathbf{L}^T$ with the difference in operand (D and $d - D$), the next operation will generate RTM and LSRTM image with the residual data to see if the resolution will get better. For RTM, the simulated data d is acquired from the forward modeling on smooth velocity model. The test setting is as same as that of Figure 4.6.

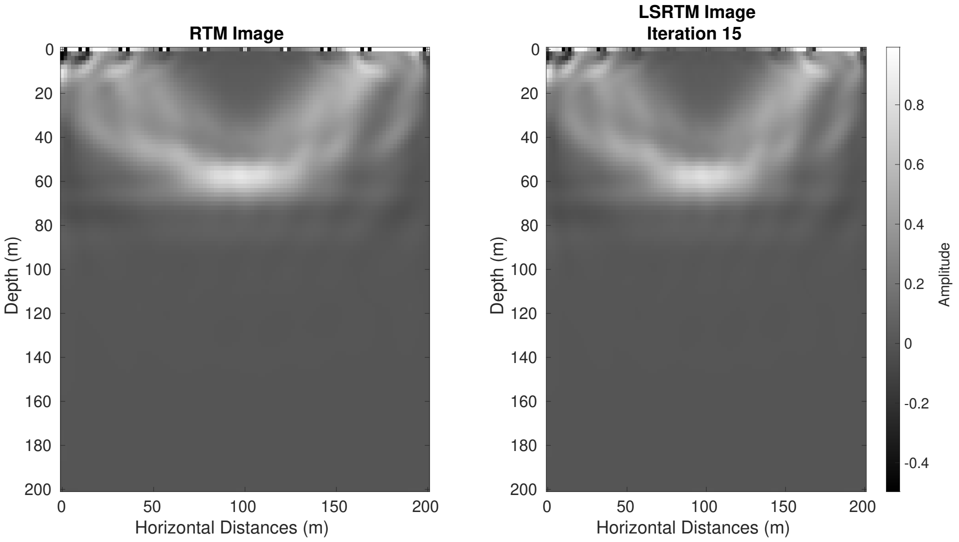


Figure 4.8: Migrated images from the LSRTM method (RTM image from residual data) on two-layered model with 10 20-Hz sources

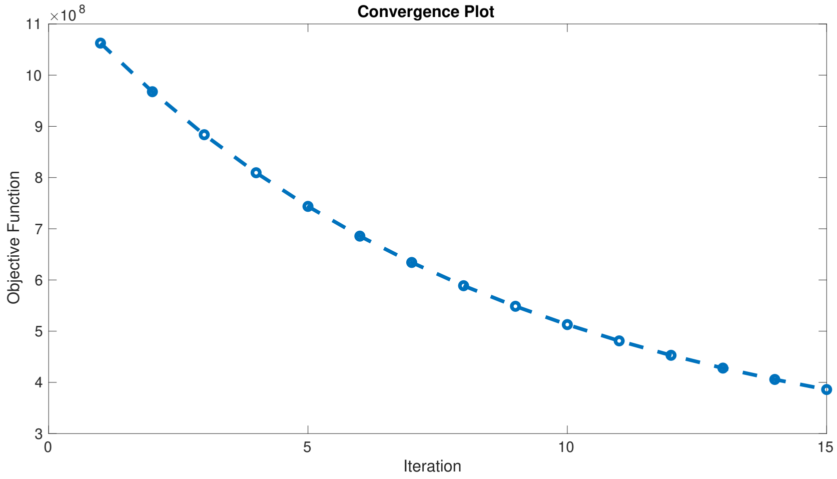
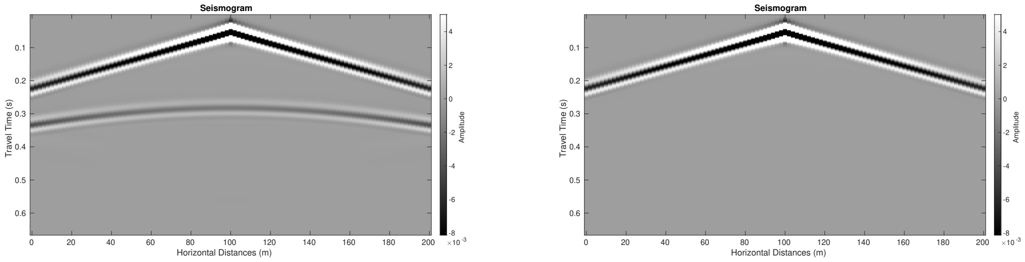
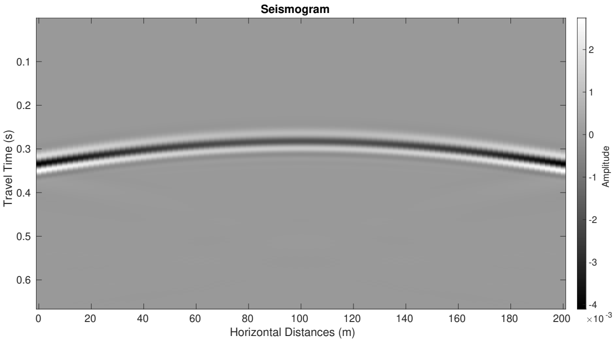


Figure 4.9: Residual graph of LSRTM (RTM image from residual data) on two-layered model with 10 20-Hz sources

The noise on the surface is siginificantly reduced on both RTM and LSRTM images in Figure 4.8, since , but the optimization is also effective only on the surface. The residual graph in Figure 4.9 still reduced exponentialy but much less steeply that that of Figure 4.7. The next experiment will focus on how to make the optimization improve the resolution at the reflectivity. A hypothesis is the optimization tries to dilute the noise on the surface because the direct wave is too strong. Therefore, the next set of results will be operated with muted direct wave on conventional LSRTM. A procedure to mute the direct wave is simple and straightforward. Another seismogram is generated on a homogeneous model that has the wave velocity equal to that of the first layer of test model. That seismogram should consist only direct wave that will be used to delete direct wave from the real seismogram. This procedure was tested on two-layered model with one source, and the results are shown in Figure 4.10.



(a) Seismogram of two-layered model (b) Seismogram of homogeneous model



(c) Seismogram with muted direct wave of two-layered model

Figure 4.10: Procedure to mute the direct wave from seismogram of two-layered model

The test setting of this experiment is still as same as that of Figure 4.6. The numerical results from LSRTM method with muted direct wave exhibit an improvement in reducing artifact. Unlike images from the last two trial, the standing-out amplitude is on the reflector position as seen in Figure 4.11. However, the same problem still stands which is the optimization still operated mainly on the surface. Despite this issue, the convergence plot in Figure 4.12 still show an exponential decrease.

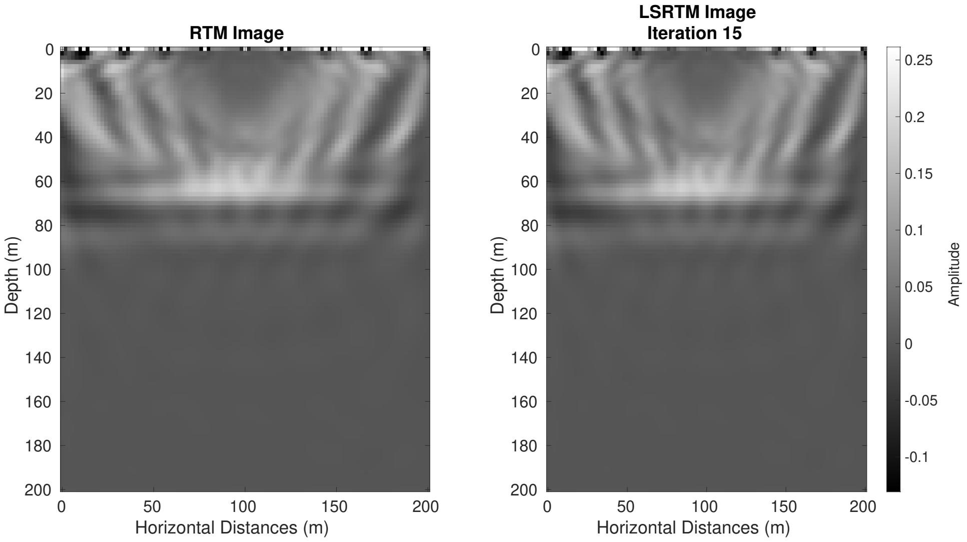


Figure 4.11: Migrated images from the LSRTM method (muted direct wave) on two-layered model with 10 20-Hz sources

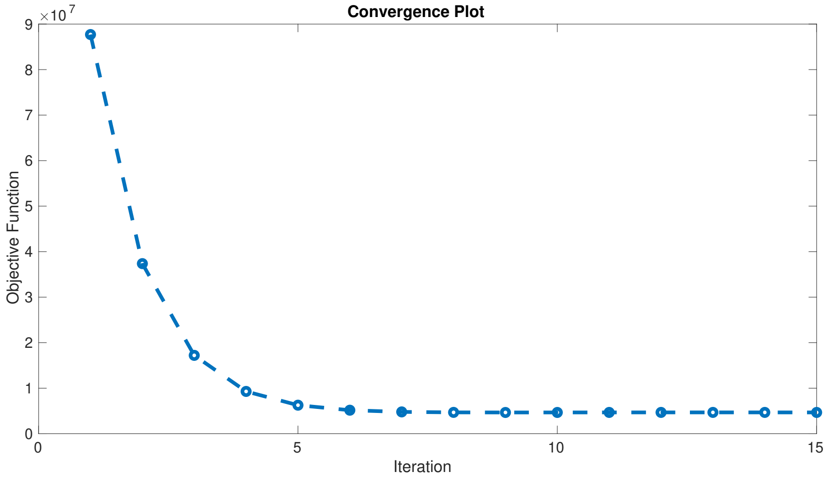


Figure 4.12: Residual graph of LSRTM (muted direct wave) on two-layered model with 10 20-Hz sources

Next, the extreme measure will be taken to make the inversion scheme optimize the reflectivity. That procedure is to mute every thing on and below the surface. The test still conducted on the two-layered model with ten sources, while ten grids at the top of the model will be muted from RTM image and during the inversion scheme. The results shown in Figure 4.13 display the improvement of reflectivity image by the inversion scheme. Due to the inteference during the inversion, the residual in Figure 4.14 is almost linear.

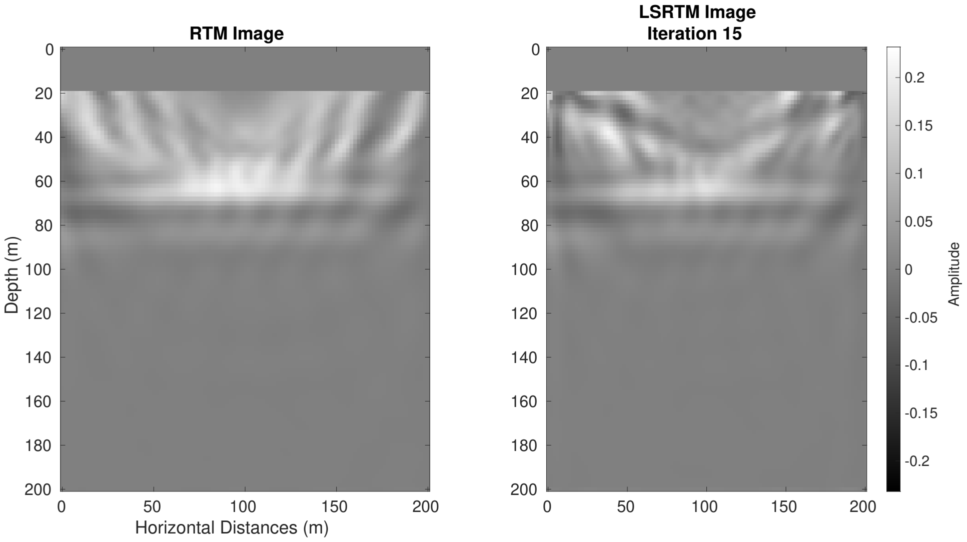


Figure 4.13: Migrated images from the LSRTM method (muted surface) on two-layered model with 10 20-Hz sources

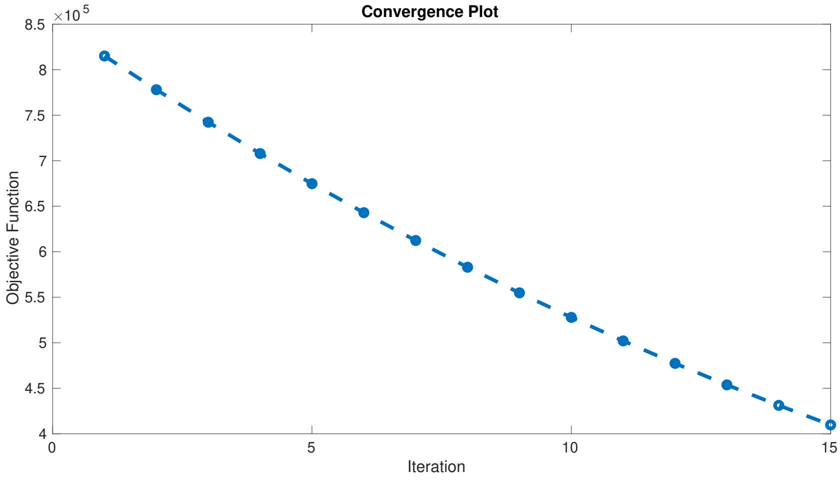


Figure 4.14: Residual graph of LSRTM (muted surface) on two-layered model with 10 20-Hz sources

4.3 Marmousi Model

Marmousi model was created by the Institut Français du Pétrole (IFP) in 1988, which its geometry is based on a profile through the North Quenguela trough in the Cuanza basin. This model is used as a test for seismic imaging and inversion scheme due to its complexity and high velocity contrast. Marmousi model size is 9.2 kilometers in horizontal distances and 3 kilometers in depth. The velocity model of Marmousi used in this work is divided into 461×151 mesh with the grid size of 20 meters. The PML boundary still consists of 20 grids for each side. A P-wave velocity model of Marmousi and its smoothed version are shown in Figure 4.15.

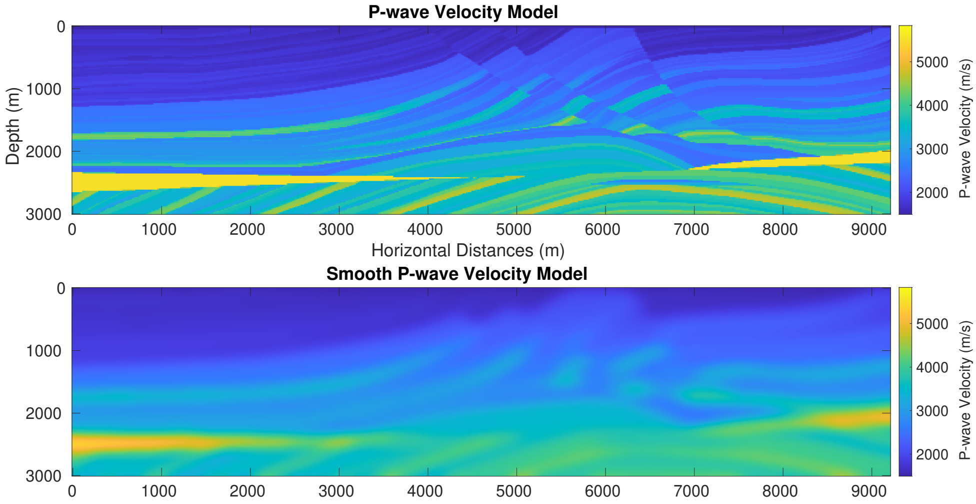


Figure 4.15: P-wave velocity model (left) and smooth P-wave velocity model (right) of Marmousi model

First, a seismogram is displayed in Figure 4.16. Compare to the seismogram of two-layered model, the seismogram of Marmousi model is much more complex. Interpretation of the structure from the seismogram is out of the question, but we can still detect the direct wave, lots of reflected waves, and boundary reflection.

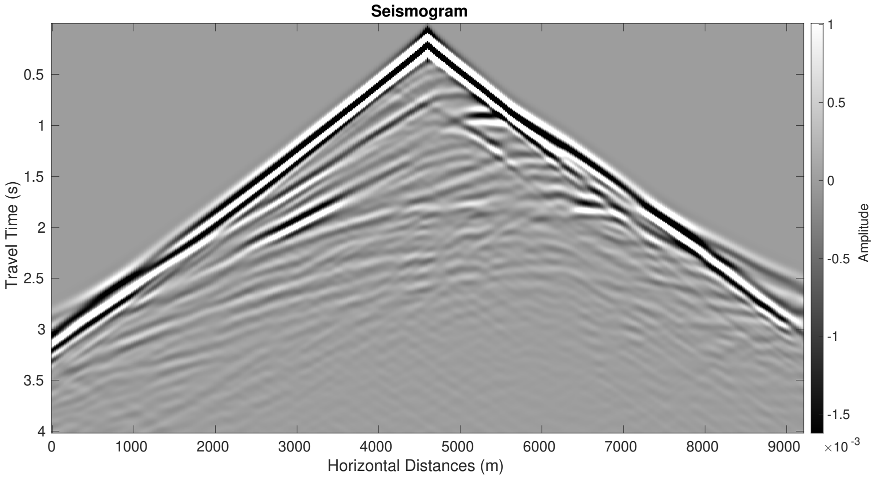


Figure 4.16: Seismogram of Marmousi model

Next, the LSRTM algorithm was operated with thirty-one 5-Hz sources equally distributed on the surface. The deominant frequency of the source is toned down to 5 Hz due to the numeral dispersion limit. The RTM image on the left panel of Figure 4.17 is able to display the bent layers, faults, and syncline structure, but the surface is tainted by the noise from sources. The inversion scheme is able to remove those noise and illuminate the structure below. However, the unwanted crosscorrelation noise is boomed up, and hindered the structure below the surface. The convergence plot in Figure 4.18 displays the exponential reduction of residual data along the iteration.

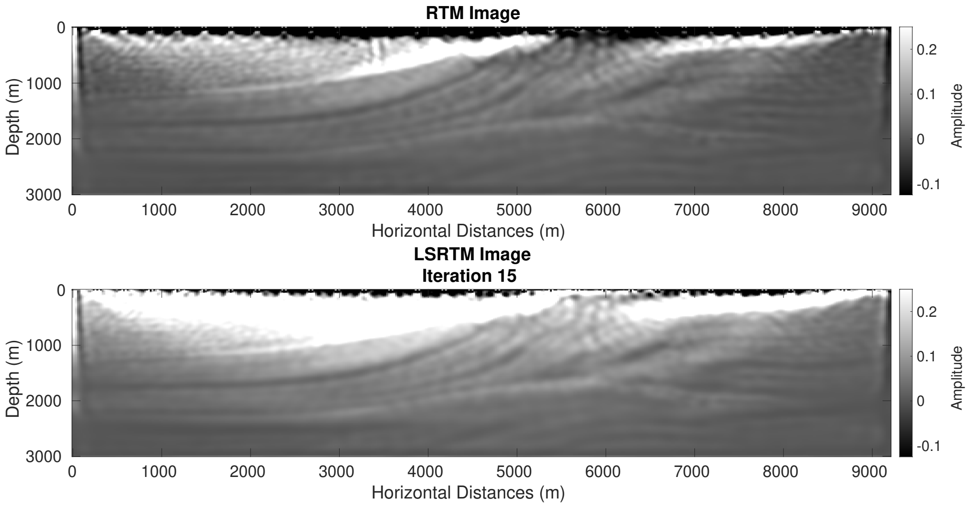


Figure 4.17: Migrated images from the LSRTM method on Marmousi model with 31 5-Hz sources

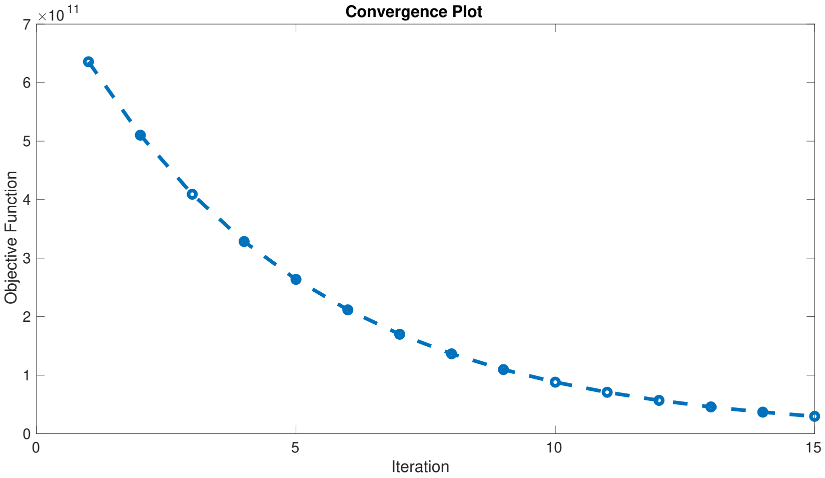


Figure 4.18: Residual graph of LSRTM on Marmousi model with 31 5-Hz sources

Similar to the experiment on two-layered model, the LSRTM algorithm with RTM image from the residual data and muted direct wave were tested on Marmousi model. Figure 4.19 and 4.20 show the numerical results from the LSRTM algorithm that generates RTM image from the residual data. The structure is more illuminated than that of the conventional LSRTM, but the surface is heavily tainted by the noise. Those noise is reduced on the optimized image, but the 15 iterations of inversion loop is not enough for diluting it. Looking at the convergence plot, the residual value could go lower with more iteration.

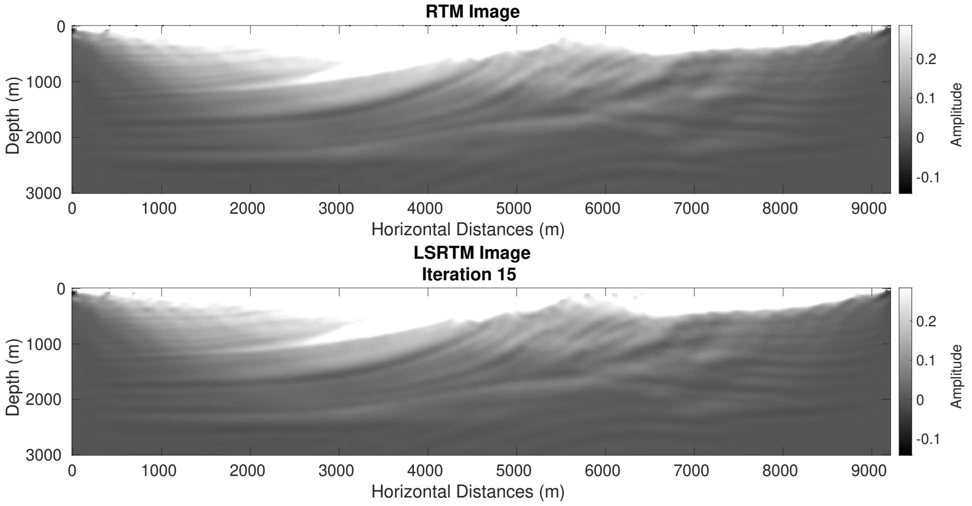


Figure 4.19: Migrated images from the LSRTM method (RTM image from residual data) on Marmousi model with 31 5-Hz sources

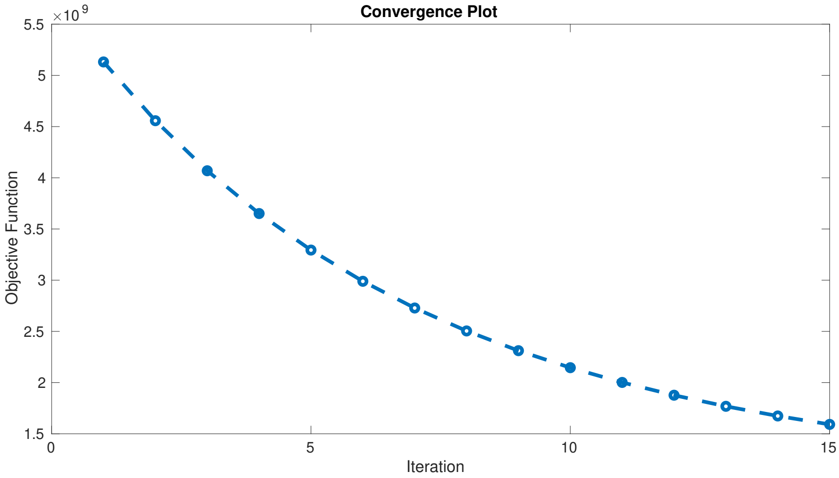
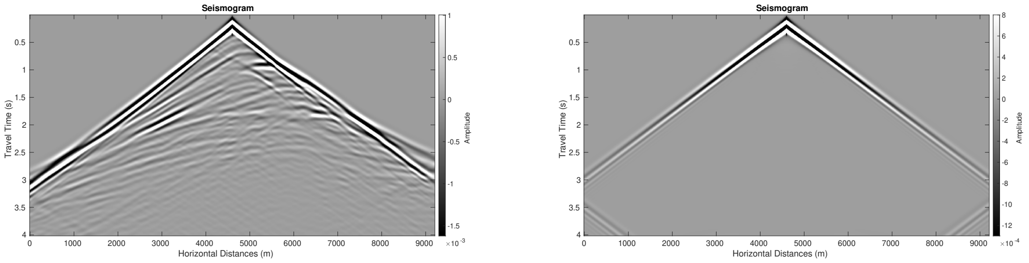
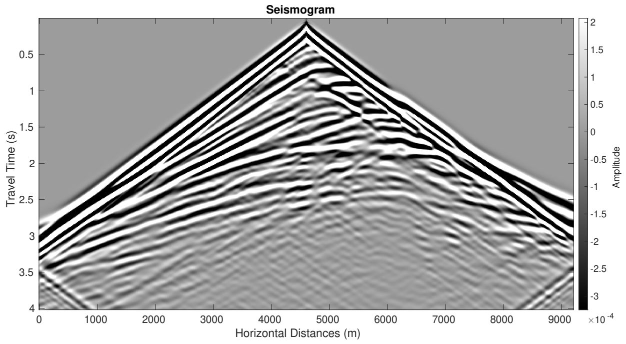


Figure 4.20: Residual graph of LSRTM method (RTM image from residual data) on Marmousi model with 31 5-Hz sources

Seeing how the image around surface contains lots of noise, muting direct wave should improve its resolution. In consequences, the LSTRM algorithm with muted direct wave was applied. Figure 4.21 shows that the dominance of reflect waves are exerted. The results in Figure 4.22 are met with the expectation since the surface is much less tainted with the noise, and rhe less noise results in more vivid structure. Despite a similarity between RTM and LSRTM image, the residual value in Figure 4.23 still reduced exponentially.



(a) Seismogram of Marmousi model (b) Seismogram of homogeneous model



(c) Seismogram with muted direct wave of Marmousi model

Figure 4.21: Procedure to mute the direct wave from seismogram of Marmousi model

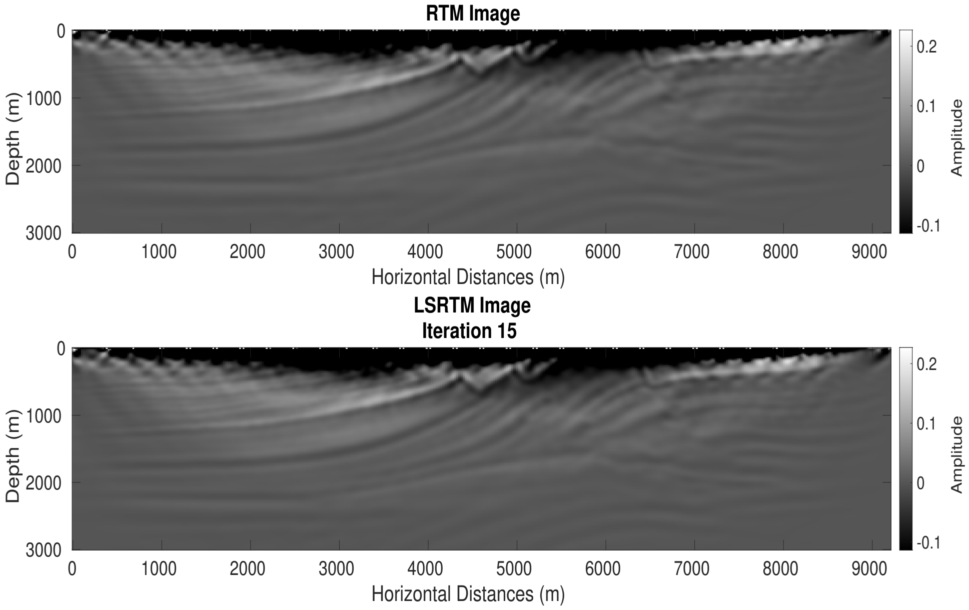


Figure 4.22: Migrated images from the LSRTM method (muted direct wave) on Marmousi model with 31 5-Hz sources

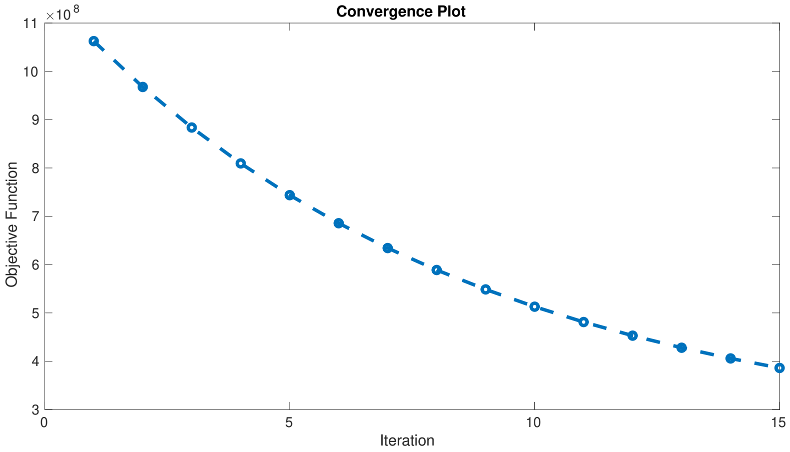


Figure 4.23: Residual graph of the LSRTM method (muted direct wave) on Marmousi model with 31 5-Hz sources

Next, an acquisition setting consists of 101 5-Hz sources evenly distributed along the surface is placed. A goal of this setting is to test the improvement of image resolution and reduction of noise around the surface through the addition of sources. As expected, Figure 4.24 exhibits more resolution with lower noise. The structure below also sharpen up by the inversion scheme. The sharpen structure gets more noticable when the images are filtered by the differential filter in Figure 4.25. Applying differential filter will remove the low frequency noise from the image which will make the structure stands out more. With 101 sources, the residual value in each iteration shown in Figure 4.26 is steeply reduced unlike that of the 31 sources. A change of LSRTM image during the inversion scheme in Figure 4.27 emphasizes the quick converge of the residual graph. The first iteration of LSRTM image already has a noticable difference from RTM image, but the image in iteration 4 alters the image even more. From iteration 4 and forth, there is a little change on image. The only spottable change is the reducing noise which is on the surface.

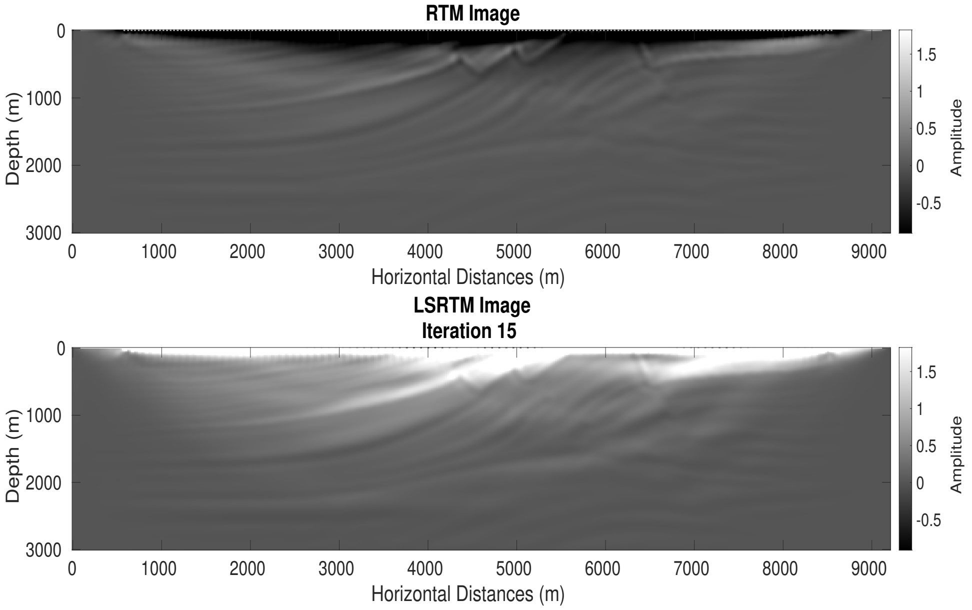


Figure 4.24: Migrated images from the LSRTM method on Marmousi model with 101 5-Hz sources

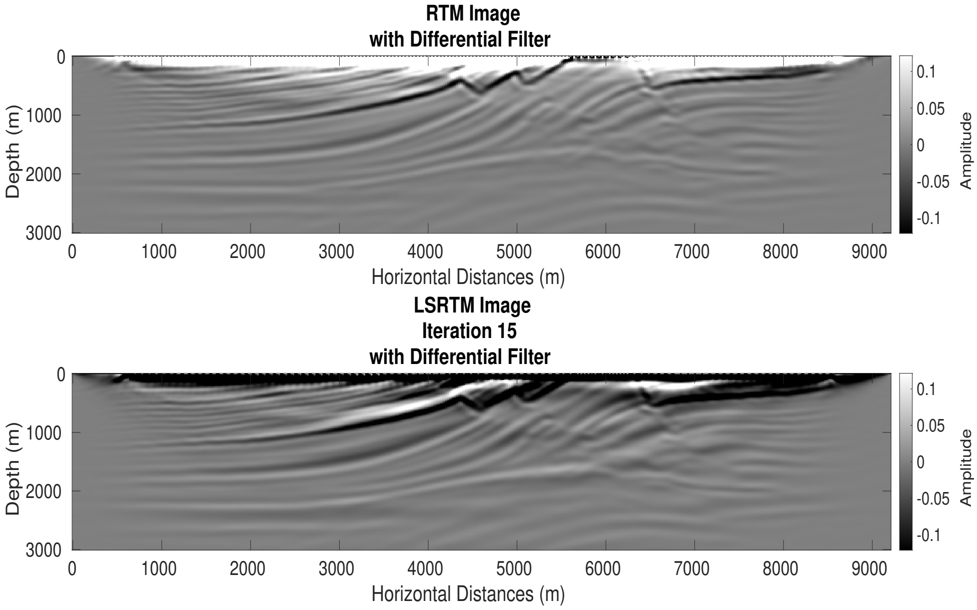


Figure 4.25: Migrated images with differential filter from the LSRTM method on Marmousi model with 101 5-Hz sources

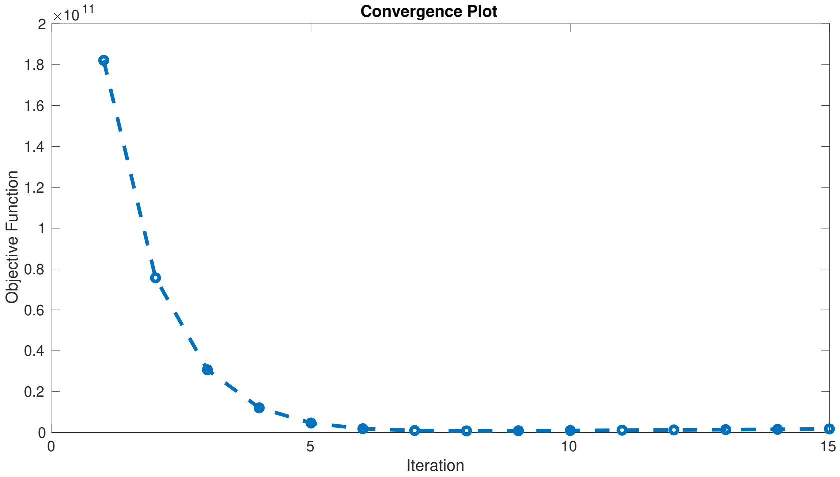


Figure 4.26: Residual graph of the LSRTM method on Marmousi model with 101 5-Hz sources)

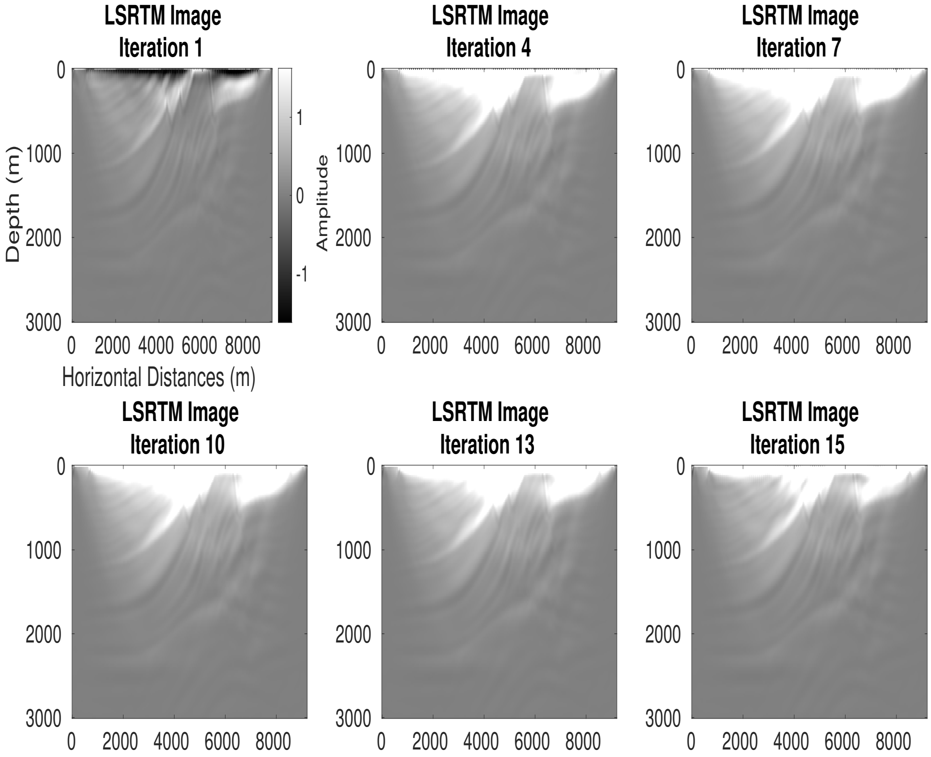


Figure 4.27: A change of migrated image during the inversion scheme

The identical problem from migrated images of Marmousmodel is the 5-Hz source is too broad to cover the complexity of Marmousi model. However, 5 Hz is the maximum frequency to be used without the occurance of grid dispersion effect in this setting. Hence, a finer velocity model is needed. This velocity model of Marmousi has 921×301 mesh with the grid size of 10 meters. The PML boundary still consists of 20 grids for each side. Now, the higher frequency is allowed with 10 Hz. The LSRTM algorithm was operated with thirty-one sources equally distributed on the surface. Both RTM and LSRTM images in Figure 4.28 show much sharper of geologic structure. A habit of inversion scheme is still similar to the images of 5-Hz source,but with the lower noise on the surface and easier distinquish of image improvement. The sharper and refined structure is confirmed by the differential filtering in Figure 4.29, while the convergence plot still bear the same characteristic.

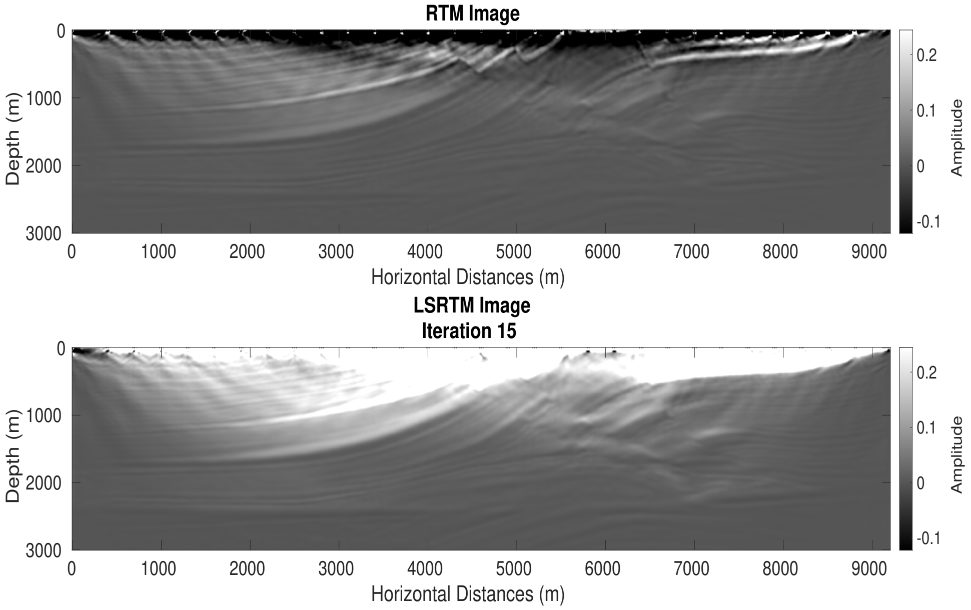


Figure 4.28: Migrated images from the LSRTM method on Marmousi model with 31 10-Hz sources

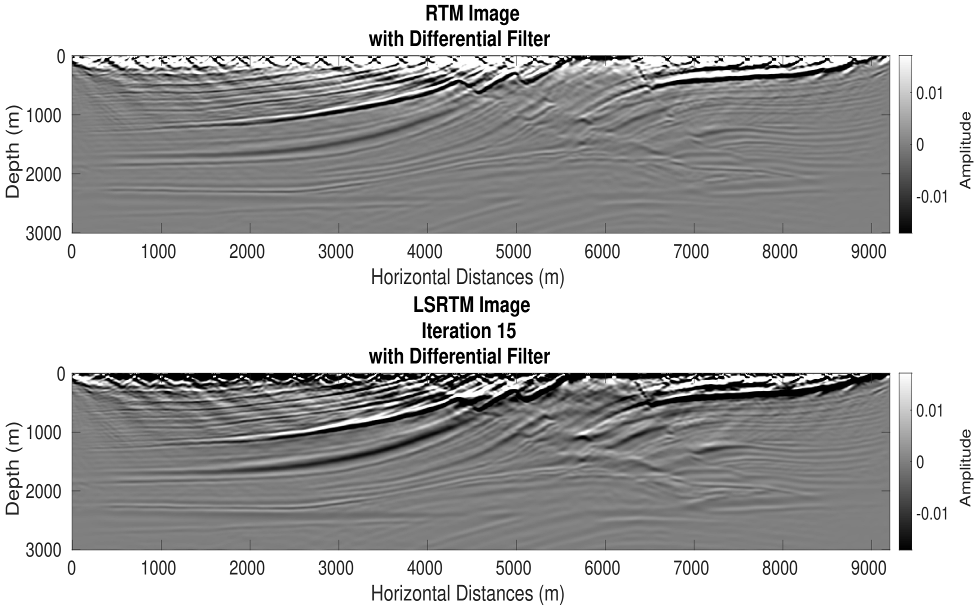


Figure 4.29: Migrated images with differential filter from the LSRTM method on Marmousi model with 31 10-Hz sources

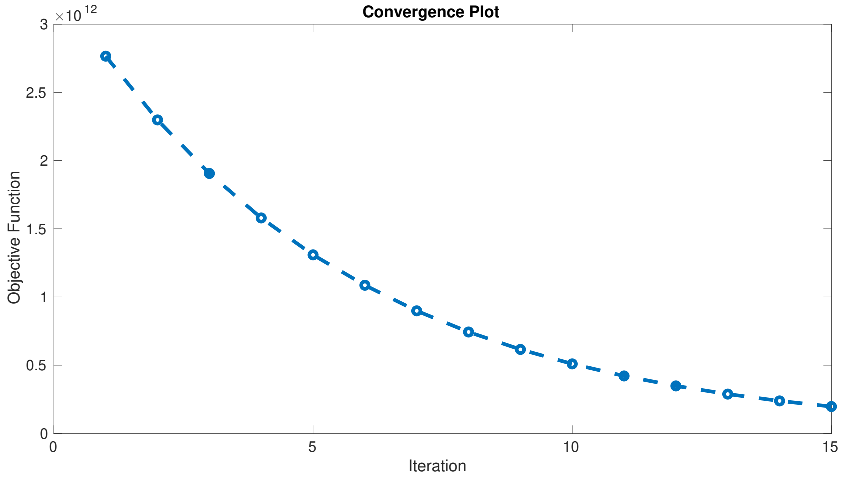
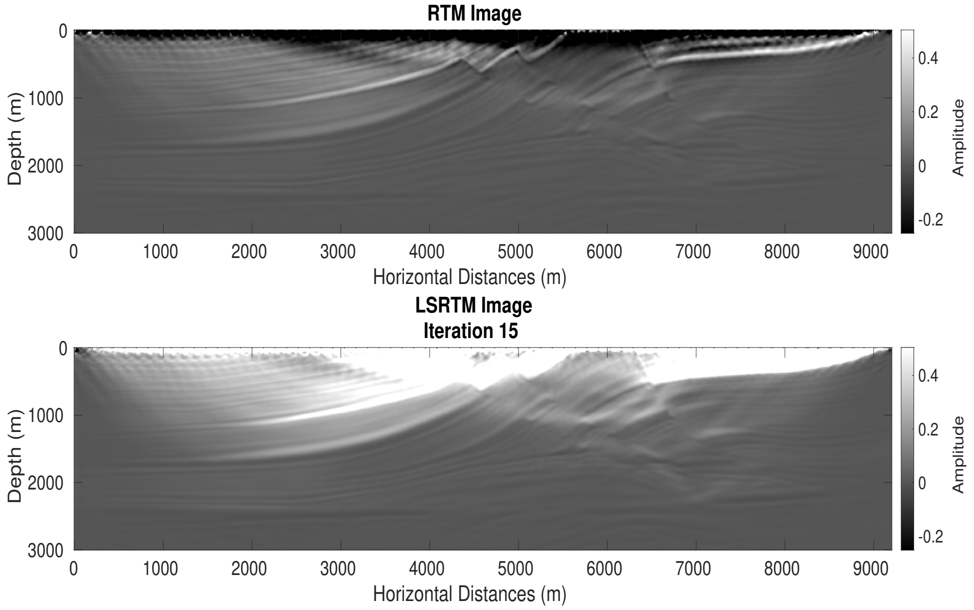


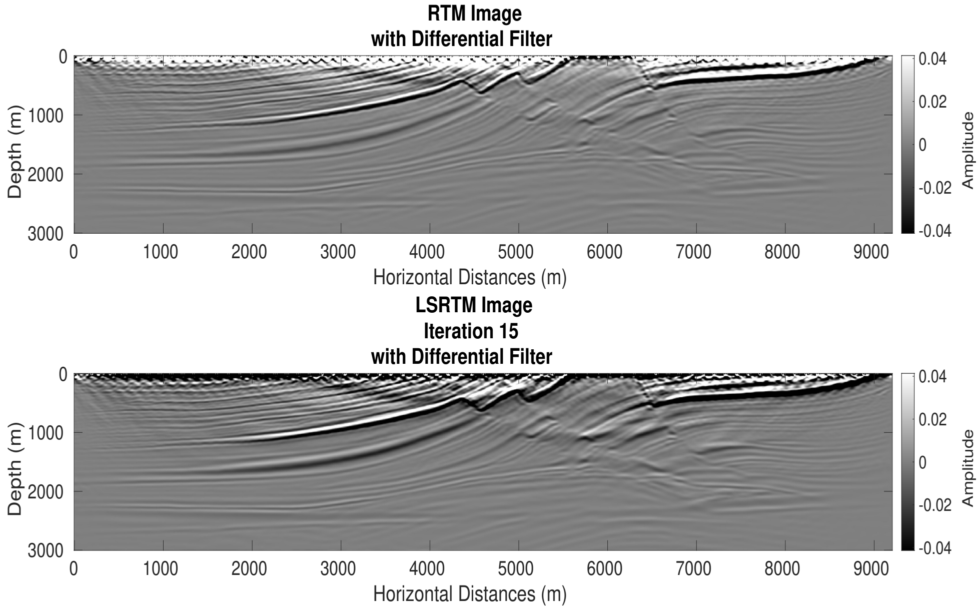
Figure 4.30: Residual graph of the LSRTM method on Marmousi model with 31 10-Hz sources

A better image resolution is expected when more sources are added in the process. Figure 4.31 shows more vivid structure and less noise compare to the migrated images of 31 10-Hz sources (Figure 4.28). The clear and accurate structure is shown after applying the differential filter to the migrated images in Figure 4.32. The convergence rate in Figure 4.33 is also steeper compare to that of the 31 sources due to more accurate data from sources additional. The residual data during the inversion scheme is also shown in Figure 4.34. Notice that the residual data is getting lower and lower in each iteratio which indicates the better fit between observe and simulated data.


RTM Image

LSRTM Image
Iteration 15

Figure 4.31: Migrated images from the LSRTM method on Marmousi model with 71 10-Hz sources


RTM Image
with Differential Filter

LSRTM Image
Iteration 15
with Differential Filter

Figure 4.32: Migrated images with differential filter from the LSRTM method on Marmousi model with 71 10-Hz sources

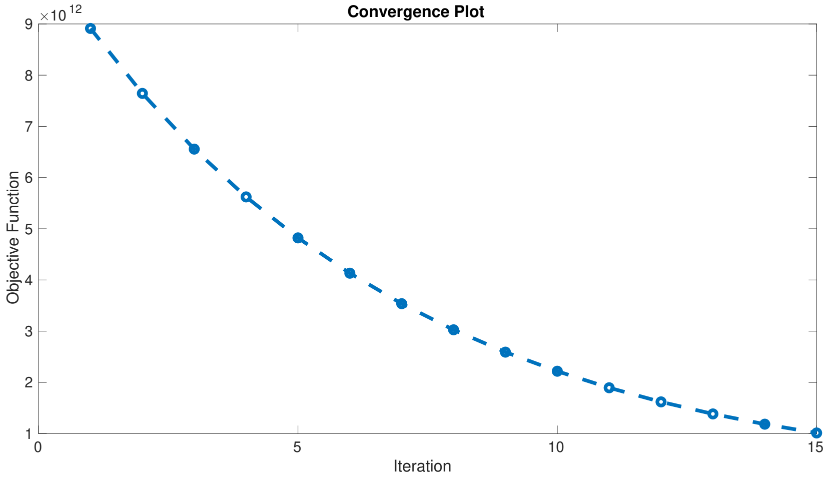


Figure 4.33: Residual graph of the LSRTM method on Marmousi model with 71 10-Hz sources

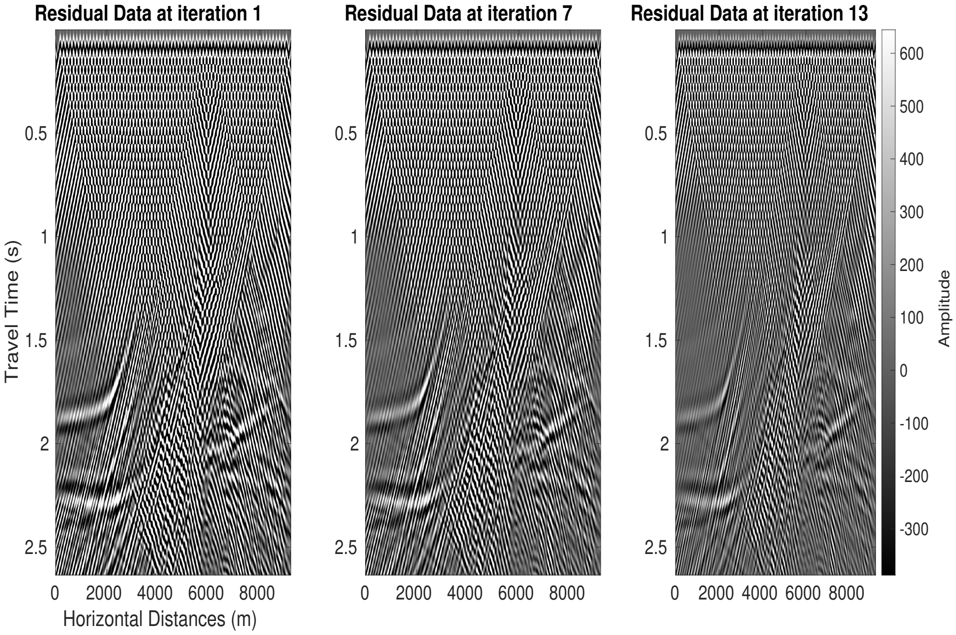


Figure 4.34: Residual data of the Marmousi model with 71 10-Hz sources in each iteration

4.4 Speedup Factor

An efficiency of GPU implementation can be measured by a speedup factor. The speedup factor enumerates how much faster is GPU processing than that of CPU which can be formulated into

$$\text{Speedup factor} = \frac{\text{CPU runtime}}{\text{GPU runtime}} \tag{4.1}$$

There are two processes that are measured which are FD calculation and fifteen iterations of LSRTM algorithm. The hardware of this test consists of Intel(R) Core (TM) i7-7700 HQ CPU @ 2.80 GHz and NVIDIA GeForce GTX 1050 Ti. The FD calculation and LSRTM algorithm were tested on different numbers of data and the speedup factors of each operation are measured as shown in Figure 4.35 and 4.36. The acquired speedup factor is around 14-15 times for FD calculation and 12-13 times for LSRTM algorithm. The beginning speedup factor is quite low due to the numbers of data is too small to emphasize the GPU efficiency compares to that of CPU. The lower speedup factor of LSRTM algorithm campares to FD calculation can be caused by the communication of host and device during the progress.

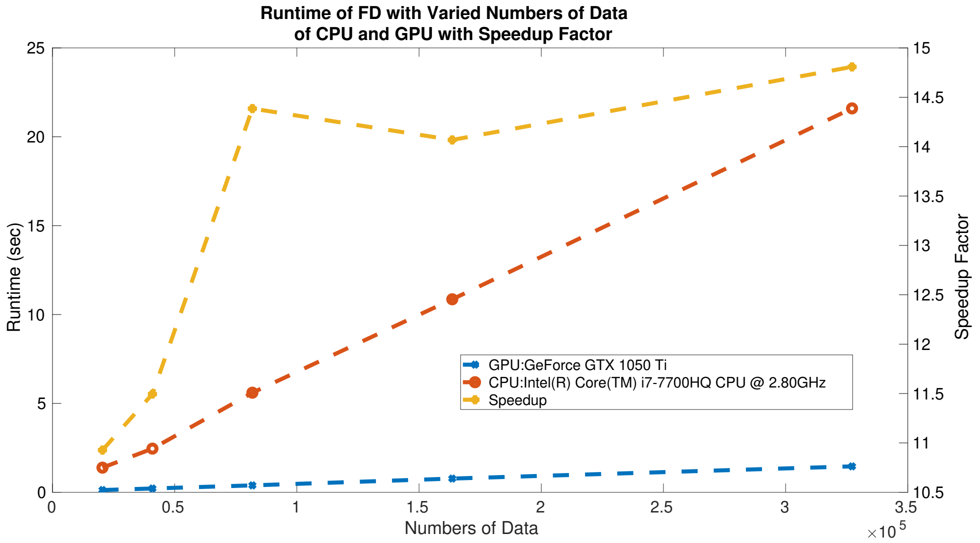


Figure 4.35: The runtime of FD calculation with varied numbers of data of CPU (Orange line) and GPU (Blue line) with the speedup factor in each problem size (Yellow line)

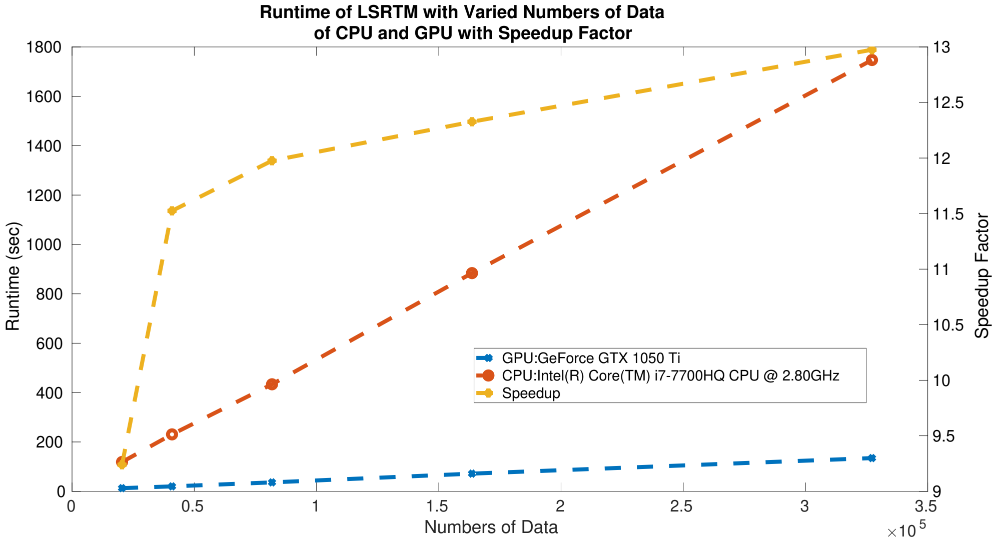


Figure 4.36: The runtime of LSRTM metohd with varied numbers of data of CPU (Orange line) and GPU (Blue line) with the speedup factor in each problem size (Yellow line)

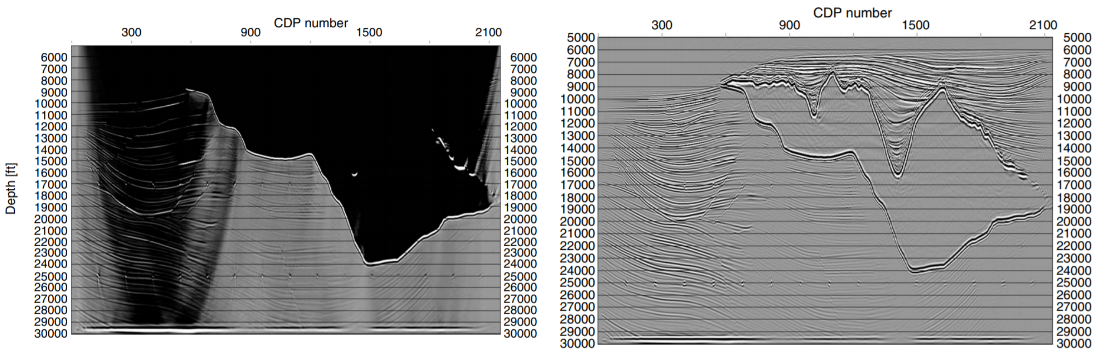
CHAPTER V
CONCLUSION

5.1 Conclusion

Least-square reverse time migration (LSRTM) method is one of the reflection seismic imaging method that improve the migrated image by applying least-squares inversion to reverse time migration (RTM) method. By computing the gradient of objective function with the adjoint-state method, LSRTM can be viewed as applying RTM method iteratively. One of the drawbacks of LSRTM method is the high computational cost, and we tried to solve this problem by implement LSRTM algorithm with GPU programming by CUDA API. Our developed LSRTM algorithm is capable of imaging a complex subsurface structure but the image around the surface is tainted by the noise. The extra implementation of up-down separation should diminish those noise below the surface which may lead to the improvement of image quality. The inversion scheme in LSRTM algorithm reduced the value of objective function exponentially, and made the geologic structure more vivid. With the additional of kernel function in GPU implementation, the total runtime of GPU-based LSRTM is about 12-to-13 times faster than that of CPU-based LSRTM for single GPU implementation.

5.2 Possible Improvement

There two possible improvement of this work which are reducing the artifact around the surface for better image resolution, and implementation of multiple GPUs to deal with a larger size of data. Liu et al. (2011) stated that since the receiver wavefield shared a path with the source wavefield, the conventional imaging condition would surely produce an artifact. However, there is a point where the directions of source and receiver wavefields contradict each other, and that point is the reflector. Therefore, the proposed imaging condition of Liu et al. (2011) only consist of the crosscorrelation of the downward propagating of source wavefield and the upward propagation of receiver wavefields and vice versa. The comparison between conventional and proposed imaging condition is shown in Figure 5.1. The noise elimination from the work of Liu et al. (2011) is worth to try for better migrated image resolution in our work.



(a) (b)

Figure 5.1: RTM image with (a) conventional imaging condition and (b) Liu et al. (2011) imaging condition from Liu et al. (2011)

With larger problem size, more resources are needed to operared. Weiss and Shragge (2013) proposed an effective way to implement multiple GPUs in FD calculation by using CUDA's P2P communication instead of MPI-based communication. Weiss and Shragge (2013)'s $ewef2d$ grants 10 times speedup, whereas $ewefd3d$ yields 16 times speedup for single GPU and 28 times speedup for two GPUs.

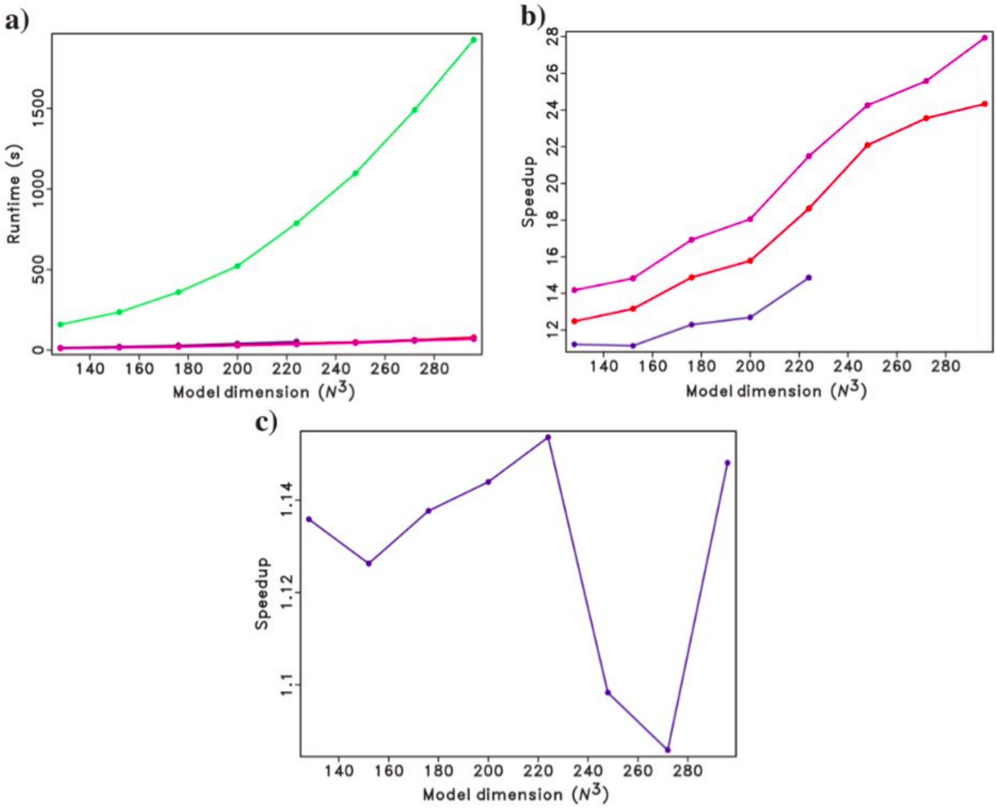


Figure 5.2: Performance metrics showing the mean of ten trials for various cube (N^3) model domains using the *ewefd3d* code. (a) Computational run time for CPU (green line), a single GPU (blue line), two GPUs with MPI communication (red line), and two GPU with P2P communication (magenta line). (b) Speedup relative to CPU for a single GPU (red line), and two GPUs with MPI (blue line) and P2P communication (magenta line). (c) Relative speed for the P2P versus MPI communication.

REFERENCES

Abdelkhalek, R., Calandra, H., Coulaud, O., Roman, J., and Latu, G. (2009). Fast seismic modeling and reverse time migration on a gpu cluster. In *2009 International Conference on High Performance Computing Simulation*, page 36–43.

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

Berenger, J.-P. (1994). A perfectly matched layer for the absorption of electromagnetic waves. *Journal of Computational Physics*, 114(2):185–200.

Bourgeois, A., Jiang, B. F., and Lailly, P. (1989). Linearized inversion: a significant step beyond pre-stack migration. *Geophysics*, 99:435–445.

Chang, W.-F. and McMechan, G. A. (1986). Reverse-time migration of offset vertical seismic profiling data using the excitation-time imaging condition. *Geophysics*, 51(1):67–84.

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

Colino, F. and Tsogka, C. (2001). Applicaton of the perfectly matched absorbing layer model to the linear elastodynamic problem. *Geophysics*, 66(1):294–307.

Dai, W. and Schuster, G. T. (2013). Plane-wave least-squares reverse-time migration. *Geophysics*, 78(4):S165–S177.

Dussaud, E., Symes, W., Williamson, P., Lemaistre, L., Singer, P., Denel, B., and Cherrett, A. (2008). Computational strategies for reverse-time migration. *SEG Technical Program Expanded Abstracts*, 27.

Dutta, G. and Schuster, G. T. (2014). Attenuation compensation for least-squares reverse time migration using the viscoacoustic-wave equation. *Geophysics*, 79(6):S251–S262.

Etgen, J., Gray, S. H., and Zhang, Y. (2009). An overview of depth imaging in exploration geophysics. *Geophysics*, 74(6):WCA5–WCA17.

Feng, Z. and Schuster, G. T. (2017). Elastic least-squares reverse time migration. *Geophysics*, 82(2):S143–S157.

Guitton, A., Kaelin, B., and Biondi, B. (2007). Least-squares attenuation of reverse-time-migration artifacts. *Geophysics*, 72(2):S19–S23.

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

Liu, F., Zhang, G., Morton, S. A., and Leveille, J. P. (2011). An effective imaging condition for reverse-time migration using wavefield decomposition. *Geophysics*, 76(1):S19–S39.

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

Micikevicius, P. (2009). 3D finite difference computation on GPUs using CUDA. In *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, page 79–84.

Neill, S. P. and Hashemi, M. R. (2018). *Fundamentals of Ocean Renewable Energy: Generating Electricity from the Sea*. Academic Press.

NVIDIA (2018). *CUDA Programming Guide, 2.1*. NVIDIA. `http://developer.download.nvidia.com/compute/cuda/2_1/toolkit/docs/NVIDIA_CUDA_Programming_Guide_2.1.pdf`.

Plessix, R.-E. (2006). A review of the adjoint-state method for computing the gradient of a functional with geophysical applicatons. *Geophys. J. int.*, 167:495–503.

Tarantola, A. (1984). Linearized inversion of seismic reflection data. *Geophysical Prospecting*, 32(6):998–1015.

Virieux, J. (1986). P-SV wave propagation in heterogeneous media: Velocity-stress finite-difference method. *Geophysics*, 51(4):889–901.

Weiss, R. and Shragge, J. (2013). Solving 3D anisotropic elastic wave equations on parallel GPU devices. *Geophysics*, 78:7–F15.

Yang, P. (2015). A graphics processing unit implementation of time-domain full-waveform inversion. *Geophysics*, 80(3):F31–F39.

Yang, P., Gao, J., and Wang, B. (2014). RTM using effective boundary saving: A staggered grid GPU implementation. *Computers & Geosciences*, 68:64–72.

Zhou, M. (2004). A well-posed PML absorbing boundary condition for 2D acoustic wave equation.

BIOGRAPHY

NAME	Mr. Phudit Sombutsirinun
DATE OF BIRTH	1 February 1995
PLACE OF BIRTH	Bangkok, Thailand
INSTITUTIONS ATTENDED	Chiang Mai University, 2013–2017 Bachelor of Science (Physics) Mahidol University, 2017–2021 Master of Science (Physics)
	Mahidol University
SCHOLARSHIP	Scholarship for Young Scientist Faculty of Science, Mahidol University, 2017
HOME ADDRESS	3 Trasvong Road, Fah Ham Subdistrict, Meuang district, Chiang Mai, 50000
E-MAIL	s.phudit1995@gmail.com