Dept. of Communications Engineering

Normal Moveout of Seismic Signals for Distributed Velocity Estimation in Subsurfaces

Master Thesis

submitted by

Zhengyu Chen

Bremen, July 7, 2021

Universität Bremen

Normal Moveout of Seismic Signals for Distributed Velocity Estimation in Subsurfaces

Dept. of Communications Engineering

Fachbereich 1 - Physics and Electronical Engineering
Institute for Telecommunications and High-Frequency Techniques (ITH)
Department of Communications Engineering
P.O. Box 33 04 40
D-28334 Bremen

Supervisor:	Dr. Ban-Sok Shin, Shengdi Wang
First Examiner:	Prof. Dr.-Ing. Armin Dekorsy
Second Examiner:	Dr. Dmitriy Shutin

I ensure the fact that this thesis has been independently written and no other sources or aids, other than mentioned, have been used.

Bremen, July 7, 2021

. . . .

Contents

Chapter 1

Introduction

1.1 Motivation

The detection of underground subsurface structures and propagation velocity of seismic waves are significant to planet exploration missions to have a profound understanding towards physical properties of planet geological structures. For instance, the unknown natural gas and mine resources could be detected and consumed to guarantee human normal life and investigation about interior earth structure also reveals how earth is formed by analyzing propagation velocity of waves via these exploration missions [2]. This kind of detection and analyzing process usually involves employment of a sensor network to measure travelling waves. In classical geophysical subsurface explorations, a typical method to acheive it is to use centralized Normal-moveout (NMO) technique that measure travel time of reflection wave from reflective interfaces [2], which requires different sensors would always transmit data at a central node and this central node executes analysis based on these local data, which is time-consuming and computational expensive in the case of large-scale sensor network. Because only the central node have the ability to do main data procssing and obtain detected results. Furthermore, for future extra-planet exploration missions, the direct control and monitoring of robots that equipped with these sensors in a centralized processing manner is heavily delayed due to remote distance between earth and other planets. Therefore, recently seismic exploration with swarm is motivated to design distributed NMO algorithms that allows robots autonomously and coorperatively explore geological subsurfaces and estimate their structures and propagation velocity of seismic waves through them in a fully distributed fashion [16]. Then, robots could work in parallel to reduce exploration time spend on subsurface detection tasks considerablely.

1.2 Problem formulation and background

In realistic scenarios, the interior earth is separated by different reflective interfaces that unevenly distributed which would form layers in between. The sesimic waves would have different propagation velocity in these different layers, which is determined by material density form these layers in terms of sand, rock, etc [2]. Additionally, waves would be reflected and refracted when they touch these interfaces and anisotropicity of layers would cause different wave propagation velocity within same layer, as they are not homogeneous medium [2]. In order to simplify the problem, the layer-cake model is introduced to replace aforementioned complicated layer structures with horizontal and homogenous layers. The basic problem addresses in this thesis is how to estimate thickness of layers and also wave propagation velocity within them. Here we provide a solution of this problem is to inject some impulse at centered-oriented source on the ground and then measure wave amplitude temporal variation with two symmetrically positioned receiver array shown in Fig 1.1. Then layer thickness and wave propagation velocity could be estimated by normal moveout from these seismic measurement after some postprocessing.

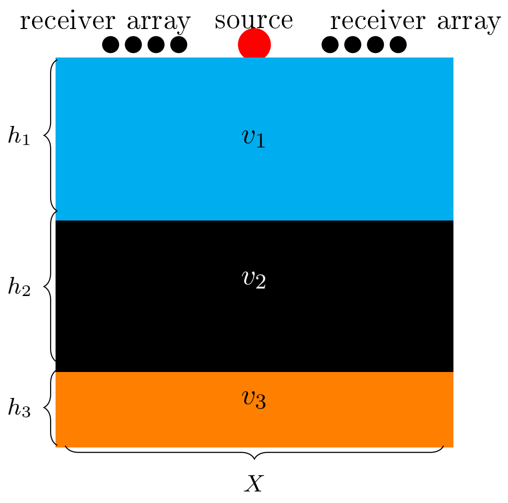


Figure 1.1: 3-layer subsurface model

1.3 Overview

The whole thesis is organized as follows: the chapter 2 incorporates the concept of seismic wave equation and provides a thereotical background of synthetic seismic data generation proceeding with numerical implementation of Finite difference time domain (FDTD) method [11]. In the chapter 3, denoising and deconvolution of noisy synthetic seismic measurment are introduced to pick travel time of reflected waves detected at sensors. Then, the chapter 4, as the main body of this thesis, provides three approaches of Normal-moveout and evaluate and compare them with simulations to realize previously mentioned estimation tasks: centralized NMO, distributed NMO with average consensus and distributed NMO with adapt-then-combine.

Chapter 2

Synthetic seismic measurement generation

The core estimation task of subsurfaces and wave propagation velocity within layer between them with Normal-moveout rely on measurement obtained at different receivers. Thus it is necessary to generate synthetic seismic measurement from analysis point of view, which involves solving seismic wave equation in a numerical way with Finite difference time-domain method or Finite element method [11]. In this thesis, Finite difference time-domain method is chosen due to simpler implementations, compared with Finite element one.

2.1 Seismic wave equation

The seismic wave equation reveals how the seismic wave propagating within a medium. As the main interest of question is how the wave propagating through 2D layered earth meidium vertical cutplane, the seismic wave acoustic model for a P-type elastic seismic body wave in 2D isotropic solid media is introduced:

$$\frac{\partial^2 \mathbf{a}(x,y,t)}{\partial x^2} + \frac{\partial^2 \mathbf{a}(x,y,t)}{\partial y^2} = \frac{\rho}{\varphi_1 + 2\varphi_2} \frac{\partial^2 \mathbf{a}(x,y,t)}{\partial t^2} = \frac{1}{c^2} \frac{\partial^2 \mathbf{a}(x,y,t)}{\partial t^2} \tag{2.1}$$

Where φ_1 and φ_2 are material-dependent quantities in the stress-strain relationship, which are called first and second Lame coefficients. ρ is linear density of the material that wave is propagating through, c is wave propagation velocity in the medium. The derivation of (2.1) can be found in Appendix. As the spatial variation of density is negligible compared with φ_1 and φ_2, we remain it as constant. Also, $\mathbf{a}(x, y, t)$ represents the wave vector at coordinate (x, y) at time t, which contains a vertical and horizontal wave component:

$$\mathbf{a}(x,y,t) = [a_h(x,y,t), a_v(x,y,t)]^T \tag{2.2}$$

From observation of (2,1), we discover that it is second order Partial differential equation (PDE), which involves derivatives with respect to mulitple variables. It also shows an equality as the divergence of wave vector is equal to its second order time derivative multiplied with a factor of $\frac{1}{c^2}$.

2.2 Numerical solution of wave equation with Finite difference

First of all, the goal is to solve wave equation in (2.3):

$$\frac{\partial^2 \mathbf{a}(x,y,t)}{\partial x^2} + \frac{\partial^2 \mathbf{a}(x,y,t)}{\partial y^2} = \frac{1}{c^2}\frac{\partial^2 \mathbf{a}(x,y,t)}{\partial t^2} \tag{2.3}$$

Subject to initial constraints:

$$\frac{\partial \mathbf{a}(x,y,0)}{\partial t} = 0, \quad \mathbf{a}(x,y,0) = [g(x,y), g(x,y)]^T \tag{2.4}$$

$$x \subset [0,X], y \subset [0,H] \tag{2.5}$$

(2.4) indicate initial velocity of wave is zero at initial time instant and initial wave amplitude is given by a function of $g(x,y)$. To obatin a numerical solution of seismic wave equation, the FDTD method is favourable due to its easy implementation. The core idea of this methodology is to decompose a finite computational region of interest into small cells and a finite simulation time into small time intervals and then solve the wave equation for each cell at given time instant individually. This procedure is clearly shown in Fig 2.1, where the left-hand side is a typical horizontal layer earth model with 3 layers and the right-hand side is deomposition of this finite layered earth model.

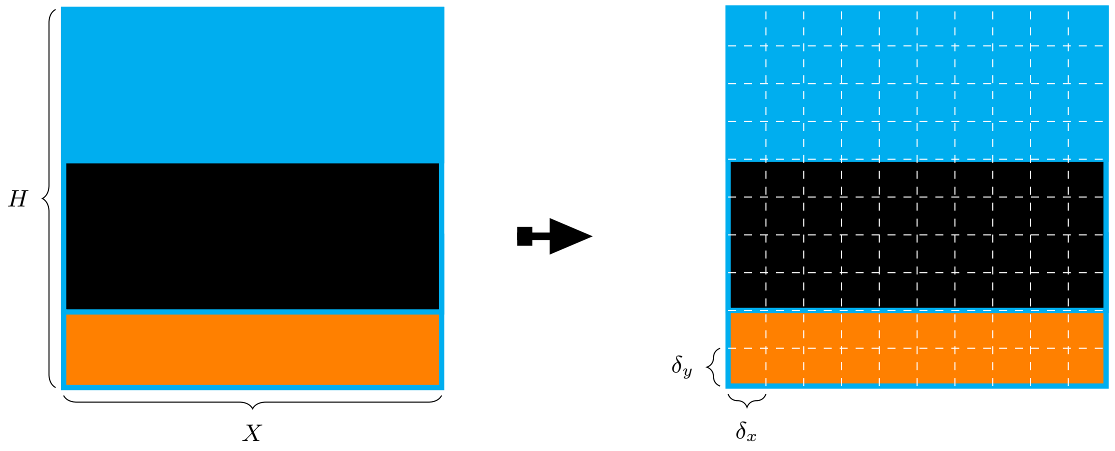


Figure 2.1: Decomposition of computational domain with FDTD

As it shown in Fig 2.1, our computational region is 2 dimensional plane defined by total length X and total layer depth H. Then, the time and spatial coordinates for each grid cell could be written as follows after discretization of FDTD:

$$x_m = m\delta x \quad m = 0, 1 \dots, M \tag{2.6}$$

$$y_n = n\delta y \quad n = 0, 1, \dots N \tag{2.7}$$

$$t_j = j\delta t \quad j = 0, 1, \dots J \tag{2.8}$$

Firstly, we consider the whole rectangular computational domain is homogenous medium and thus wave propagation velocity in different layers is same. Then the wave displacement of a given cell is annotated as $\mathbf{a}(x_m, y_n, t_j) = \mathbf{a}^j_{m,n}$. The second order spatial derivatives in horizontal x direction and vertical y direction could be approximated with central difference [11] as shown in (2.9) and (2.10):

$$\frac{\partial^2 \mathbf{a}(x,y,t)}{\partial y^2} \approx \frac{\mathbf{a}(x, y+\delta y, t) - 2\mathbf{a}(x,y,t) + \mathbf{a}(x, y-\delta y, t)}{\delta y^2} \tag{2.9}$$

$$\frac{\partial^2 \mathbf{a}(x,y,t)}{\partial x^2} \approx \frac{\mathbf{a}(x+\delta x, y, t) - 2\mathbf{a}(x,y,t) + \mathbf{a}(x-\delta x, y, t)}{\delta x^2} \tag{2.10}$$

For second order time derivatives:

$$\frac{\partial^2 \mathbf{a}(x,y,t)}{\partial t^2} \approx \frac{\mathbf{a}(x, y, t+\delta t) - 2\mathbf{a}(x,y,t) + \mathbf{a}(x, y, t-\delta t)}{\delta t^2} \tag{2.11}$$

Then, replace finite difference approximation given in (2.9) and (2.10) and (2.11) into orignal seismic wave PDE in (2.3), after rearrangement we get:

$$\mathbf{a}^{j+1}_{m,n} \approx r_1^2(\mathbf{a}^j_{m+1,n} - 2\mathbf{a}^j_{m,n} + \mathbf{a}^j_{m-1,n})) + 2\mathbf{a}^j_{m,n} - \mathbf{a}^{j-1}_{m,n} + r_2^2(\mathbf{a}^j_{m,n+1} - 2\mathbf{a}^j_{m,n} + \mathbf{a}^j_{m,n-1})) \tag{2.12}$$

Where r_1 and r_2 are given as:

$$r_1 = \frac{c^2 \delta t^2}{\delta x^2} \tag{2.13}$$

$$r_2 = \frac{c^2 \delta t^2}{\delta y^2} \tag{2.14}$$

One point should be noted is so far phase velocity of seismic wave in (2.12) is considered as constant, which should be modified after incorporation of multiple layer model. From observation of (2.12), it can be seen this numrical approximation follows a recursion procedure that reveals wave vector for an individual cell at time step $j+1$ is associated with wave vector of its neighbouring cells at current time step j and its wave vector at current and previous time step. This is the reason why this method is called Finite difference time-domain method, as it involves recursive time step update in the algorithm.

Extend to multiple-layer model

However, practically speaking, propagation velocity of wave velocity in different layer is different due to different geological material. Therefore, it is necessary to incorporate varying velocity into discretization of PDE with FDTD. The velocity of grid cell is given as discrete variable $c_{m,n}$, Firstly, we have wave PDE in 2D as:

$$\frac{\partial^2 \mathbf{a}(x_m, y_n, t_j)}{\partial x^2} + \frac{\partial^2 \mathbf{a}(x_m, y_n, t_j)}{\partial y^2} = \frac{1}{c_{m,n}^2} \frac{\partial^2 \mathbf{a}(x_m, y_n, t_j)}{\partial t^2} \tag{2.15}$$

Then we could write left-hand expression as:

$$c_{m,n}^2 \frac{\partial^2 \mathbf{a}(x_m, y_n, t_j)}{\partial x^2} + c_{m,n}^2 \frac{\partial^2 \mathbf{a}(x, y, t)}{\partial y^2} = \frac{\partial (q_{m,n} \frac{\partial \mathbf{a}(x_m, y_n, t_j)}{\partial x})}{\partial x} + \frac{\partial (q_{m,n} \frac{\partial \mathbf{a}(x_m, y_n, t_j)}{\partial y})}{\partial y} \tag{2.16}$$

Where $q_{m,n} = c_{m,n}^2$, then consider variable within bracket and define:

$$\phi = q_{m,n} \frac{\partial \mathbf{a}(x_m, y_n, t_j)}{\partial x} \tag{2.17}$$

$$\rho = q_{m,n} \frac{\partial \mathbf{a}(x_m, y_n, t_j)}{\partial y} \tag{2.18}$$

It turns out the second order derivative on the left-hand side of expression (2.16) can be written as:

$$q_{m,n} \frac{\partial^2 \mathbf{a}(x_m, y_n, t_j)}{\partial x^2} + q_{m,n} \frac{\partial^2 \mathbf{a}(x_m, y_n, t_j)}{\partial y^2} = [\frac{\partial \phi}{\partial x}]_{m,n}^j + [\frac{\partial \rho}{\partial y}]_{m,n}^j \tag{2.19}$$

Then we would solve derivatives with central difference and get:

$$[\frac{\partial \phi}{\partial x}]_{m,n}^j \approx \frac{\phi_{m+\frac{1}{2},n}^j - \phi_{m-\frac{1}{2},n}^j}{\delta x} \tag{2.20}$$

$$[\frac{\partial \rho}{\partial y}]_{m,n}^j \approx \frac{\rho_{m,n+\frac{1}{2}}^j - \rho_{m,n-\frac{1}{2}}^j}{\delta y} \tag{2.21}$$

Recall (2.17) and (2.18) and continue with calculation of algorithmic mean:

$$\phi_{m+\frac{1}{2},n}^j \approx \frac{1}{2}(q_{m,n} + q_{m+1,n})(\frac{\mathbf{a}_{m+1,n}^j - \mathbf{a}_{m,n}^j}{\delta x}) \tag{2.22}$$

$$\phi_{m-\frac{1}{2},n}^j \approx \frac{1}{2}(q_{m,n} + q_{m-1,n})(\frac{\mathbf{a}_{m,n}^j - \mathbf{a}_{m-1,n}^j}{\delta x}) \tag{2.23}$$

$$\rho_{m,n+\frac{1}{2}}^j \approx \frac{1}{2}(q_{m,n} + q_{m,n+1})(\frac{\mathbf{a}_{m,n+1}^j - \mathbf{a}_{m,n}^j}{\delta y}) \tag{2.24}$$

$$\rho^j_{m,n-\frac{1}{2}} \approx \frac{1}{2}(q_{m,n} + q_{m,n-1})(\frac{\mathbf{a}^j_{m,n} - \mathbf{a}^j_{m,n-1}}{\delta y}) \tag{2.25}$$

Then we replace above four formulas into (2.20) and (2.21) and combine with (2.15) and finally reach:

$$\begin{aligned}\mathbf{a}^{j+1}_{m,n} \approx\ & d_1^2(\frac{1}{2}(q_{m,n} + q_{m+1,n})(\mathbf{a}^j_{m+1,n} - \mathbf{a}^n_{m,j}) - \frac{1}{2}(q_{m,n} + q_{m-1,n})(\mathbf{a}^j_{m,n} - \mathbf{a}^j_{m-1,n}))\\ &+ d_2^2(\frac{1}{2}(q_{m,n} + q_{m,n+1})(\mathbf{a}^j_{m,n+1} - \mathbf{a}^j_{m,n}) - \frac{1}{2}(q_{m,n} + q_{m,n-1})(\mathbf{a}^j_{m,n} - \mathbf{a}^j_{m,n-1}))\\ &+ 2\mathbf{a}^j_{m,n} - \mathbf{a}^{j-1}_{m,n}\end{aligned} \tag{2.26}$$

Where $d_1 = \frac{\delta x}{\delta t}$ and $d_2 = \frac{\delta y}{\delta t}$ respectively. The numerical approximation in (2.26) provides a general numerical solution regarding to wave equation with repect to multiple layer model with homogenous layers. Interestingly, for cells within one homogenous layer, discrete velocity variable $q_{m,n}$ is same for its neighboring cells and therefore (2.26) will reduce to original numerical solution shown in (2.12). However, for grid cells at the interfaces betweeen layers, it would not be same which implies wave propagation pattern would be different in these interfaces compared the case within one layer. Additionally, it should be noted (2.26) is only valid for grid cells inside computational region. As for the cells at the boundary, it is related to calculation of wave vector for cells beyond computation region, which are also called ghost cells. However, their calculation could be solved by defining boundary condition illustrated in the next section.
For the initialization of FDTD, solution in (2.26) should be modified, as once we replace $j = 0$ into (2.26), we would encounter wave vector $\mathbf{a}^{-1}_{m,n}$ at time step -1. This makes the problem seems hard to solve, but fortunately we could recall initial condition in (2.4) draw a conclusion with central difference:

$$\frac{\partial \mathbf{a}^0_{m,n}}{\partial t} \approx \frac{\mathbf{a}^1_{m,n} - \mathbf{a}^{-1}_{m,n}}{2\delta t} = 0 \tag{2.27}$$

Then we get $\mathbf{a}^1_{m,n} = \mathbf{a}^{-1}_{m,n}$ and solution at second time step could be formulated as:

$$\begin{aligned}\mathbf{a}^{1}_{m,n} \approx\ & d_1^2(\frac{1}{4}(q_{m,n} + q_{m+1,n})(\mathbf{a}^0_{m+1,n} - \mathbf{a}^n_{m,j}) - \frac{1}{4}(q_{m,n} + q_{m-1,n})(\mathbf{a}^0_{m,n} - \mathbf{a}^0_{m-1,n}))\\ &+ d_2^2(\frac{1}{4}(q_{m,n} + q_{m,n+1})(\mathbf{a}^0_{m,n+1} - \mathbf{a}^0_{m,n}) - \frac{1}{4}(q_{m,n} + q_{m,n-1})(\mathbf{a}^0_{m,n} - \mathbf{a}^0_{m,n-1}))\\ &+ \mathbf{a}^0_{m,n}\end{aligned} \tag{2.28}$$

Numerical stability of FDTD

As FDTD involves a numerical approximation with discretization in both time and spatial domain. Not all choices of space and time discretization level woul lead a stable numerical solution. The classical condition to restriction of discretziation levels is the so called Courant-Friedrichs–Lewy (CFL) condition [10]. This is a general condition applied for numerical approximation with Finite difference in multi-dimensional perspective. The underlying principle of this condition can be explained as follows: imagine we want to measure the wave amplitude at discrete time instant with a equal duration of δt. Physically speaking, wave would spend another time denoted by δt_p travel that allows it travel to an adjacent cell with certain velocity. In order to make sure that we can capture the wave at adjacent cell, δt must be smaller or equal to δt_p. In other words, the numerical solution domain should be within the physical analytical domain. In 2D case, it is given in (2.29):

$$\frac{c_{m,n}\delta t}{\delta x} + \frac{c_{m,n}\delta_t}{\delta_y} <= C_{max} \tag{2.29}$$

Where C_{max} is called Courant number and usually set to be 1. Then we rearange (2.29) and get:

$$\delta t <= \frac{1}{\frac{c_{m,n}}{\delta x} + \frac{c_{m,n}}{\delta_y}} \tag{2.30}$$

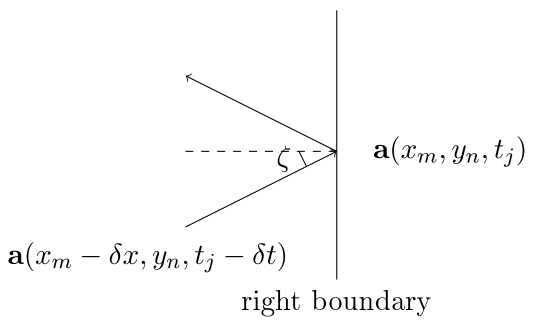
Condition in (2.30) should hold for all discrete velocity variable in the whole multiple layer earth model and thus we could set specific CFL condition as

$$\delta t <= \frac{1}{c_{m,n}^{max}(\frac{1}{\delta x} + \frac{1}{\delta_y})} \tag{2.31}$$

Where $c_{m,n}^{max}$ is maximum wave propagation velocity in the whole layered earth medium.

2.3 Absorbing and reflective boundaries

As computational region is a rectangle with specific boundaries, therefore specific boundary conditions need to be considered to calculate wave vector for boundary cells. In general, there are two types of boundary conditions: Absorbing boundary condition (ABC) and reflective boundary condition. The motivation to apply ABC is to absorb incoming wave at artificial boundary to make simulation scenario more realistic, as these artificial boundaries don't exist in real environment but we need to include them for numerical simulation analysis. Similarly, reflective boundary is used to observe reflections from boundary. The commonly used ABC includes Mur absorbing boundary condition, Perfectly matched layer (PML) [3], etc. Mur ABC is easy to implement but show imperfections in 2D and higher dimension simulations, whereas PML absorbs wave much better in 2D but computational expensive because we need to truncate an outer absorbing region with computational region and waves would be absorbed gradually from this absorbing region. In this thesis, we choose Mur ABC for simulation for left and right hand-side boundaries to reduce unwanted artifical reflections to simplify problem.



ζ
$\mathbf{a}(x_m, y_n, t_j)$
$\mathbf{a}(x_m - \delta x, y_n, t_j - \delta t)$
right boundary

Figure 2.2: incoming wave with oblique angle

2.3.1 Mur absorbing boundary condition (Mur ABC)

A very simple absorbing boundary condition is Mur ABC, which is perfect for a travelling wave with velocity c moving perpendicularly to the boundary. The idea of Mur ABC is obtained by discretizing first order derivative of wave vector $\mathbf{a}(x, y, t)$ at point $\frac{\delta x}{2}$ away from left and right boundaries, firstly we come to first order derivative of wave vector that can be derived from wave 2D wave PDE and evaluate derivative at point $(\frac{\delta x}{2}, y_n, t_j + \frac{\delta t}{2})$ for left boundary and $(x_M - \frac{\delta x}{2}, y_n, t_j + \frac{\delta t}{2})$ for right boundary respectively in (2.32)-(2.33):

$$\frac{\partial \mathbf{a}(x, y, t)}{\partial x} = -\frac{1}{c_{0,n}} \frac{\partial \mathbf{a}(x, y, t)}{\partial t} \tag{2.32}$$

$$\frac{\partial \mathbf{a}(x, y, t)}{\partial x} = -\frac{1}{c_{M,n}} \frac{\partial \mathbf{a}(x, y, t)}{\partial t} \tag{2.33}$$

Then we can get:

$$\frac{\frac{\mathbf{a}^j_{1,n} + \mathbf{a}^{j+1}_{1,n}}{2} - \frac{\mathbf{a}^j_{0,n} + \mathbf{a}^{j+1}_{0,n}}{2}}{\delta x} = -\frac{1}{c^2_{0,n}} \frac{\frac{\mathbf{a}^{j+1}_{1,n} + \mathbf{a}^{j+1}_{0,n}}{2} - \frac{\mathbf{a}^j_{0,n} + \mathbf{a}^j_{1,n}}{2}}{\delta t} \tag{2.34}$$

Finally, we calculate wave at left boundary as:

$$\mathbf{a}^{j+1}_{0,n} = \mathbf{a}^j_{1,n} + p\mathbf{a}^{j+1}_{1,n} - p\mathbf{a}^j_{0,n} \tag{2.35}$$

Where $p = \frac{c_{0,n}\delta t - \delta x}{c_{0,n}\delta t + \delta x}$. Similarly we could also calculate wave at right boundary recursively as:

$$\mathbf{a}^{n+1}_{M,j} = \mathbf{a}^n_{M-1,j} + p_1\mathbf{a}^{n+1}_{M-1,j} - p_1\mathbf{a}^n_{M,j} \tag{2.36}$$

With $p_1 = \frac{c_{M,n}\delta t - \delta x}{c_{M,n}\delta t + \delta x}$. Now imagine incoming wave is exposing boundary with an incidence angle ζ and reflected by boundary which is shown in Fig 2.2. The incident and reflected wave amplitude $a_i(x_m, y_n, t_j)$ and $a_r(x_m, y_n, t_j)$ can be calculated with complex exponential term in (2.37) and (2.38) [3]:

$$a_i(x_m, y_n, t_j) = a_i^0 \exp(i\omega t_j - ik_x x_m - ik_y y_n) \tag{2.37}$$

$$a_r(x_m, y_n, t_j) = a_r^0 \exp(i\omega t_j + ik_x x_m - ik_y y_n) \tag{2.38}$$

Where a_i^0 and a_r^0 are initial incident and reflected wave amplitude and k_x and k_y are wavenumber in x and y direction respectively. As wave field at point (x_m, y_n, t_j) is same as at point $(x_m - \Delta x, y_n, t_j - \Delta t)$ [3]. Thus, we can derive following equality as:

$$a_i(x_m, y_n, t_j) = a_i(x_m - \delta x, y_n, t_j - \delta t) \tag{2.39}$$

$$a_r(x_m, y_n, t_j) = a_r(x_m - \delta x, y_n, t_j - \delta t) \tag{2.40}$$

Combine (2.37)-(2.48), we can obtain:

$$a_i^0 \exp(i\omega t_j) + a_r^0 \exp(i\omega t_j) = a_i^0 \exp(i\omega(t_j - \delta t) + ik_x \delta x) + a_r^0 \exp(i\omega(t_j - \delta t) - ik_x \delta x) \tag{2.41}$$

Divide both sides of (2.41) with $\exp(i\omega t_j)$, we get:

$$a_i^0 + a_r^0 = a_i^0 \exp(i\omega(-\delta t) + ik_x \delta x) + a_r^0 \exp(i\omega(-\delta t) - ik_x \delta x) \tag{2.42}$$

Continue with first order expoential expansion, we get:

$$a_i^0 + a_r^0 = a_i^0(1 - i\omega\delta t + ik_x \delta x) + a_r^0(1 - i\omega\delta t - ik_x \delta x) \tag{2.43}$$

After rearrangement, the ratio between reflected wave amplitude versus incident one is given as:

$$\begin{aligned} \frac{a_r^0}{a_i^0} &= \frac{k_x - \frac{\omega}{c}}{k_x + \frac{\omega}{c}} \\ &= \frac{\cos\zeta - 1}{\cos\zeta + 1} \end{aligned} \tag{2.44}$$

The last line in (2.44) is done with assistance of wavenumber definition $k_x = \frac{\omega \cos\zeta}{c}$. Therefore, only at normal incidence angle which means $\zeta = 0$, there is no reflection from boundary. For other oblique angles, Mur ABC still produce certain portion of reflection and the larger incidence angle means more wave would be reflected back.

2.3.2 Dirichlet and Neumann conditions

The simplest reflective boundary condition is Dirichlet condition [11], which is quietly frequently used in solving ordinary and partial differential equations. For Dirichlet condition, wave amplitude in both directions are set to be 0 at boundary, which has physical meaning that there is no wave remaining at boundary and incoming wave would be totally reflected by the boundary. In mathematics, it is given as:

$$\mathbf{a}_{M,n} = \mathbf{a}_{M,0} = \mathbf{a}_{m,N} = \mathbf{a}_{m,0} = \mathbf{0} \tag{2.45}$$

The Neumann condition [11] specifies the normal derivative of wave PDE at boundary to be zero. For the ground and bottom boundaries of our computational region, normal direction follows y direction. Therefore, we could derive correponding Neumann condition for them:

$$\frac{\partial \mathbf{a}(x,0,t)}{\partial y} = \frac{\mathbf{a}_{m,1} - \mathbf{a}_{m,-1}}{2\delta y} = 0 \tag{2.46}$$

$$\frac{\partial \mathbf{a}(x,H,t)}{\partial y} = \frac{\mathbf{a}_{m,N+1} - \mathbf{a}_{m,N-1}}{2\delta y} = 0 \tag{2.47}$$

Which provide solution for ghost cells as: $\mathbf{a}_{m,1} = \mathbf{a}_{m,-1}$ and $\mathbf{a}_{m,N+1} = \mathbf{a}_{m,N-1}$. Subsequently, we return back to wave equation solution in (2.26) and solve for wave vector at ground and bottom boundaries as:

$$\mathbf{a}^{j+1}_{m,0} \approx (\frac{c_{m,0}\delta t}{\delta x})^2(\mathbf{a}^j_{m+1,0} - 2\mathbf{a}^j_{m,0} + \mathbf{a}^j_{m-1,0}) + 2\mathbf{a}^j_{m,0} - \mathbf{a}^{j-1}_{m,0} + (\frac{c_{m,0}\delta t}{\delta y})^2(\mathbf{a}^j_{m,1} - 2\mathbf{a}^j_{m,0} + \mathbf{a}^j_{m,1}) \tag{2.48}$$

$$\mathbf{a}^{j+1}_{m,N} \approx (\frac{c_{m,N}\delta t}{\delta x})^2(\mathbf{a}^j_{m+1,N} - 2\mathbf{a}^j_{m,N} + \mathbf{a}^j_{m-1,N}) + 2\mathbf{a}^j_{m,N} - \mathbf{a}^{j-1}_{m,N} + (\frac{c_{m,N}\delta t}{\delta y})^2(\mathbf{a}^j_{m,N-1} - 2\mathbf{a}^j_{m,N} + \mathbf{a}^j_{m,N-1}) \tag{2.49}$$

(2.48) and (2.49) could be further simplified in the case of choosing same discretization level in x and y directions as shown in -:

$$\mathbf{a}^{j+1}_{m,0} \approx (\frac{c_{m,0}\delta t}{\delta x})^2(\mathbf{a}^j_{m+1,0} - 4\mathbf{a}^j_{m,0} + \mathbf{a}^j_{m-1,0} + 2\mathbf{a}^j_{m,1}) + 2\mathbf{a}^j_{m,0} - \mathbf{a}^{j-1}_{m,0} \tag{2.50}$$

$$\mathbf{a}^{j+1}_{m,N} \approx (\frac{c_{m,N}\delta t}{\delta x})^2(\mathbf{a}^j_{m+1,N} - 4\mathbf{a}^j_{m,N} + \mathbf{a}^j_{m-1,N} + 2\mathbf{a}^j_{m,N-1}) + 2\mathbf{a}^j_{m,N} - \mathbf{a}^{j-1}_{m,N} \tag{2.51}$$

Physically speaking, Neumann condition implies the energy flow of seismic wave is bounded within computational region and therefore when wave hit a Neumann boundary, it would be reflected but unlike Dirichlet condition there is still some portion of wave remaining at boundary.

2.4 Verification of Finite difference implementation

As Finite difference method is an an approximation of analytic solution of 2D wave PDE. Therefore, the quality of this numerical scheme should be evaluated and investigated. To do this, we add an additional source function in the original wave PDE as:

$$\frac{\partial^2 \mathbf{a}(x,y,t)}{\partial x^2} + \frac{\partial^2 \mathbf{a}(x,y,t)}{\partial y^2} = \frac{1}{c^2}\frac{\partial^2 \mathbf{a}(x,y,t)}{\partial t^2} + \mathbf{s}(x,y,t) \tag{2.52}$$

To verify our numerical solution from FDTD, a manual solution $\mathbf{a}_r(x, h, t)$ could be constructed as shown in (2.53):

$$\mathbf{a}_r(x, y, t) = [x(1-x)(t)^2 y, x(1-x)(t)^2 y]^T \tag{2.53}$$

Then take (2.53) into (2.52), we solve for source term as:

$$\mathbf{s}(x, y, t) = [-2t^2 y - \frac{2y}{c^2}(1-x)x, -2t^2 y - \frac{2y}{c^2}(1-x)x]^T \tag{2.54}$$

Recall FDTD solution in (2.12), then the modified FDTD solution after incorporation of source term is given in (2.55):

$$\mathbf{a}^{j+1}_{m,n} = r_1^2(\mathbf{a}^j_{m+1,n} - 2\mathbf{a}^j_{m,n} + \mathbf{a}^j_{m-1,n})) + 2\mathbf{a}^j_{m,n} - \mathbf{a}^{j-1}_{m,n} + r_2^2(\mathbf{a}^j_{m,n+1} - 2\mathbf{a}^j_{m,n} + \mathbf{a}^j_{m,n-1})) - c^2\delta t^2 \mathbf{s}(x_m, y_n, t_j) \tag{2.55}$$

Initialization of FDTD is given by mannually ground truth solution:

$$\mathbf{a}^0_{m,n} = \mathbf{a}_r(x_m, y_n, 0) \tag{2.56}$$

Recall (2.28) The wave vector at second time instant is calculated as:

$$\mathbf{a}^1_{m,n} = \frac{1}{2} r_1^2(\mathbf{a}^0_{m+1,n} - 2\mathbf{a}^0_{m,n} + \mathbf{a}^0_{m-1,n})) + \mathbf{a}^0_{m,n} + \frac{1}{2} r_2^2(\mathbf{a}^0_{m,n+1} - 2\mathbf{a}^0_{m,n} + \mathbf{a}^0_{m,n-1})) - \frac{1}{2} c^2\delta t^2 \mathbf{f}(x_m, y_n, \delta t) \tag{2.57}$$

To simplify verification process, we choose a single layer with simulation setup given in table 2.1.

Table 2.1: Computation region setup for verification of FDTD

length X	**depth H**	**Simulation time**	**wave velocity c**
200 m	100 m	0.3 s	5744.23 m/s

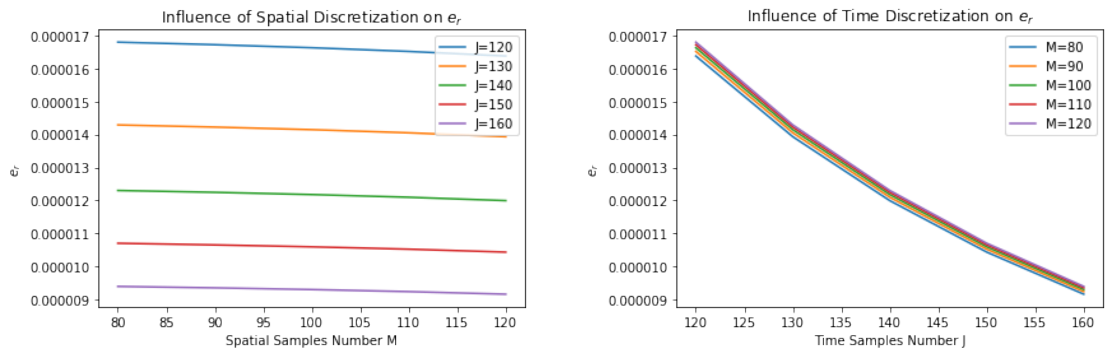
To analysis parameter influence to final approximation error we proceed with normalized error definition for single cell:

$$e^j_{m,n} = \frac{(||\mathbf{a}_r(x_m, y_n, t_j) - \mathbf{a}^j_{m,n}||)^2}{||\mathbf{a}_r(x_m, y_n, t_j)||^2} \tag{2.58}$$

The average numerical approximation error at given time instant j as metric that average over all cells inside boundary to evaluate performance of FDTD in an average sense. For boundary cells, we can specify them have same value as in the manual ground truth solution and therefore there is no numerical error for them.

$$e_r = \frac{\sum_{m=2}^{M-1}\sum_{n=2}^{N-1} e^j_{m,n}}{(M-2)\times(N-2)} \tag{2.59}$$

To observe how spatial discretization level δx and δy and time resolution δt influences numerical approximation error of FDTD, we simulate for different number of total grid cells M in horizontal direction and total time samples J for a given computation region. To ease computation, $M = N$ holds on which sure same number of cells in x and y directions. The y-axis of Fig 2.3 is numerical approximation error e_r, where x-axis represents



(a) Spatial resolution effect (b) Time resolution effect

Figure 2.3: The influence of time and spatial resolution to numerical approximation error E

number of spatial and time samples respectively. Each solid line in the plot is estimation curve for a certain choice of spatial and time samples combination (M, J). From Fig 2.3, an interesting point could be discovered as finer spatial and time discretization level will reduce numerical approximation error. This is because numerical error in FDTD arises from finite difference approximation of gradient for analytical gradient. Thus, when discretization is finer, the numrical gradient at single point will be closer to analytical gradient and thus we would get better approximation results. However, increase same amount of time samples will reduce it much more significantly than that of spatial samples. This shows discretization of hyperbolic wave PDE is more sensitive to time resolution, instead of spatial resolution. As, from classical signal processing perspective with Nyquist sampling theory. The wave amplitude pattern between two time steps might appearing aliasing and cause increasing numerial estimation error in the case of large time resolution, which equivalents a small sampling frequncy. But for spatial resolution, it involves the wave diffusion in spatial domain which doesn't have such a restriction.

2.5 Wave propagation in 3-layer model

2.5.1 Ricker wavelet

In geophysical exploration, to generate synthetic seismic measurement for further analysis, an easy approach is using a hammer strike the ground surface and simulate how seismic wave propagates through geological medium and finally observe the measurement at geophones to get a rough estimate wave propagation velocity. Therefore, the Ricker wavelet is enforced at manually positioned source and then we record wave amplitude variation as impulse response for the cells on the ground surface that represent receivers with FDTD in a numerical way and proceed further processing. Ricker wavelet is chosen for simulation as it is one of best approximation function regarding to seismic spectra among all different kinds of impulse functions in terms of differentiation and nonnegativity of approximation function etc [14]. Thereotically, Ricker wavelet is a second order derivative of Gaussian function [15] which has form:

$$r(t) = (1 - \frac{1}{2}\omega_p^2 t^2) \exp\left\{-\frac{1}{4}\omega_p^2 t^2\right\} \tag{2.60}$$

$\omega_p = 2\pi f$ is peak frequency of wavelet in radians and for simulation f is specified to 200 Hz. It is symmetric and has a peak at zero time. The peak frequency cannot be chosen too small so that its bandwidth is too large, as source excitation is like striking a hammer on the ground which is very fast procedure. Meanwhile, we cannot start a signal from peak from physical perspective, therefore this Ricker wavelet is shifted with l_{peak} positive time sample to allow it start with very small value.

We can observe it shows an overall decay-increase-converge pattern on the right-hand side of peak from Fig 2.4 (a). The spectrum of Ricker wavelet is calculated by applying Fourier transformation with respect to (2.60):

$$R(\omega) = \mathcal{F}(r(t)) = \frac{4\sqrt{\pi}\omega^2}{\omega_p^3} \exp\left(-\frac{\omega^2}{\omega_p^2}\right) \tag{2.61}$$

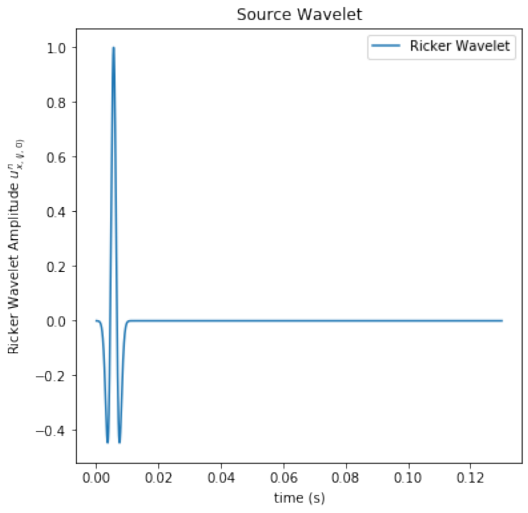
The detailed derivation of (2.61) can be seen in Appendix. Additionally, maximum of spectrum of Ricker wavelet corresponds to peak frequency of Ricker wavelet, which is observed from subplot 2.4 (b) and stated mathematically by taking derivative of (2.61) with respect to ω:

$$\begin{aligned} \frac{\partial \mathcal{F}(r(t))}{\partial \omega} &= \frac{8\sqrt{\pi}\omega}{\omega_p^3} \exp\left(-\frac{\omega^2}{\omega_p^2}\right) - \frac{4\sqrt{\pi}\omega^2}{\omega_p^3} \exp\left(-\frac{\omega^2}{\omega_p^2}\right)\frac{2\omega}{\omega_p^2} \\ &= \frac{8\sqrt{\pi}\omega}{\omega_p^3}(1 - (\frac{\omega^2}{\omega_p^2}))\exp\left(-\frac{\omega^2}{\omega_p^2}\right) \end{aligned} \tag{2.62}$$

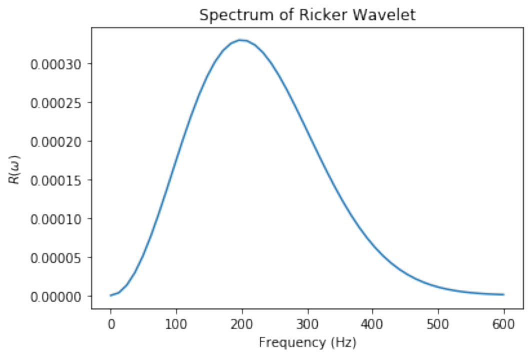
Specify (2.62) to zero for peak in the spectrum, we obtain:

$$\omega = \omega_p \tag{2.63}$$

To visualize how wave is propagating through multi-layer earth model, a 3-layer subsur-



(a) Ricker wavelet



(b) Ricker wavelet spectrum

Figure 2.4: The comparison of Ricker wavelet and its spectrum (200 Hz)

face model is introduced with corresponding subsurface parameters shown in Table 2.2. The layer velocity is chosen to be larger in the deeper layer, which follows the Snell's law

Table 2.2: Subsurface parameters setup for wave visualization

Layer number	**Layer thickness**	**Layer velocity**
1	15 m	0.667 km/s
2	29 m	1.7 km/s
3	12 m	2.2 km/s

given in (2.64) [2]: for an incident wave hit on the interface, it will be refracted to deeper layer and then we have relationship between incidence angle of two consecutive layers Θ_i and Θ_{i+1}:

$$\frac{\cos\Theta_i}{\cos\Theta_{i+1}} = \frac{v_i}{v_{i+1}} \tag{2.64}$$

Incidence angle is the angle between seismic ray and horizontal direction. Then we have $\Theta_i > \Theta_{i+1}$ and get:

$$v_i < v_{i+1} \tag{2.65}$$

Table 2.3: FDTD setup for wave visualization

δx	δy	δt	**Simulation time**
0.23333 m	0.23333 m	5e-5 s	0.25s

The discretization levels for FDTD are given in Table 2.3. There are 12 receivers placed with equal inter-receiver distance on one side of source, which means we have $D = 24$ receivers on the ground with total length of simulation region $X = 28m$. The source-to-receiver distance of those receivers are defined as $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$, with corresponding grid cell indices in FDTD as $\mathbf{I} = [m_1, m_2, \dots, m_D]^T$

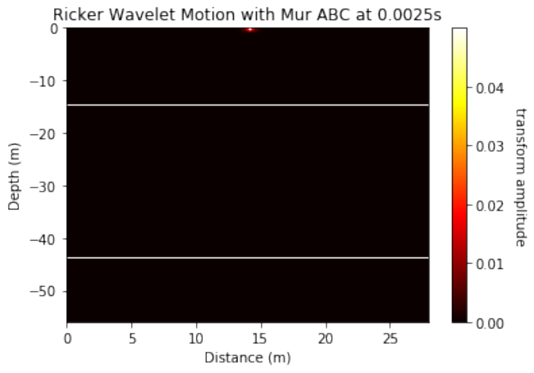
2.5.2 Wave propagation simulation

Numerical solution of wave equation by FDTD enables us to simulate wave propagation through different layers which is clearly shown with 4 subplots in Fig 2.5. For the cell represents source, wave vector is directly given by shifted Ricker wavelet with same vertical and horizontal wave components:

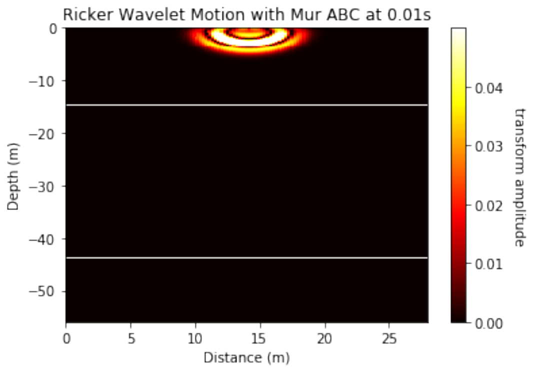
$$\begin{aligned}\mathbf{a}(x_s, 0, t_j) =&[(1 - \frac{1}{2}\omega_p^2(t_j - l_{peak}\delta t)^2)\exp\left\{-\frac{1}{4}\omega_p^2(t_j - l_{peak}\delta t)^2\right\}, \\ &(1 - \frac{1}{2}\omega_p^2(t_j - l_{peak}\delta t)^2)\exp\left\{-\frac{1}{4}\omega_p^2(t_j - l_{peak}\delta t)^2\right\}]^T\end{aligned} \tag{2.66}$$

Where x_s is x-coordinate of source and $l_{peak}\delta t$ is shift time of Ricker wavelet. For simulation, we set it as $l_{peak}\delta t = \frac{5\sqrt{2}}{\omega_p}$. As initially there is only wave exists at source location injected by Ricker wavelet in the compuational domain, other cells are initialized with a zero wave vector.

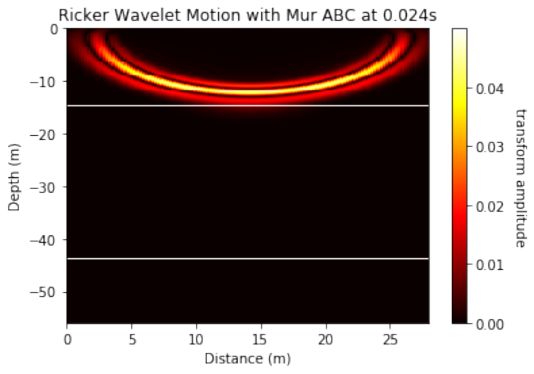
In order to show the wave pattern clearer, plotted wave amplitude shown in the colorbar of following subfigures in Fig 2.5 are limited by $0.05a_{max}$ with a_{max} is largest wave amplitude. The interfaces between layers are separated by two white horizontal lines in figures. The computational region is set to be a rectangle with dimension: 28m for length and 56m for total depth. For right-hand side, left-hand side and ground, Mur ABC is applied to absorb incoming wave and for bottom boundary, we apply Neumann condition to reflect incoming wave. However, ground surface should be a reflective boundary and bottom boundary should be an absorbing boundary in reality, as bottom boundary is an artificial boundary set for FDTD and ground surface is a hard-wall to reflect wave. But for estimation of subsurfaces, the reflection from bottom boundary is needed to allow receivers place on the ground surface measure this reflection and estimate layer thickness of last layer. Therefore, we enforce above-mentioned boundary condition for four borders of computational region. The Fig 2.5 (a) is wave pattern at very begining of simulation and shows a bright dot at center of ground line, which actually corresponds to the injection of Ricker wavelet and its expansion to neighboring cells. From comparison between Fig 2.5 (a)-(c), we find out wave is expanding in all direction circularly and then its outermost envelope touches the first interface located with depth of 15m shown in Fig 2.5 (c). At this time, wave propagation pattern is different than before as it is entering into other homogeneous medium with different material. Therefore, refraction and reflection occur at this stage, which we can observe part of wave is reflected by the interface and would be detected by receivers after returning back to the ground. The remaining part is refracted into second layer and its envelope is becoming broader than before.



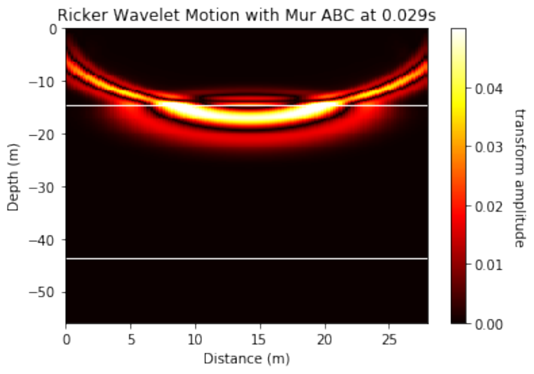
(a) $t_j = 0.0025\text{s}$



(b) $t_j = 0.01\text{s}$



(c) $t_j = 0.024\text{s}$

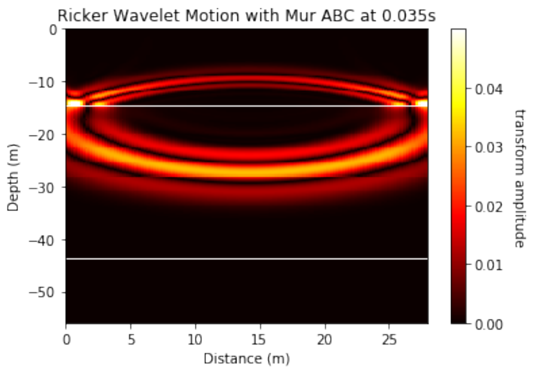


(d) $t_j = 0.029\text{s}$

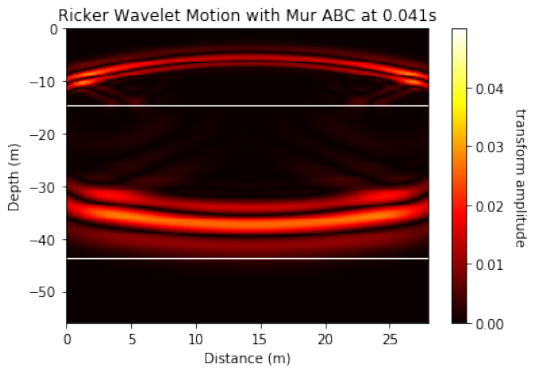
Figure 2.5: Wave propagation at different time instants

2.5.3 Imperfections with Mur ABC

Incoming wave that touches right and left boundaries at first layer has been absorbed perfectly by Mur ABC, which can be seen from comparison of Fig 2.6 (a) and Fig 2.5 (d). However, when refracted wave which is expanding in the second layer exposes right and left boundary, small portion of reflections can be observed in the Fig 2.6 (b). This



(a) $t_j = 0.035$s



(b) $t_j = 0.041$s

Figure 2.6: Small reflection from absorbing boundary

actually shows imperfection of Mur ABC in 2D FDTD simulation, as wave is no longer propagating perpendicular to right and left boundaries after refraction and therefore only perpendicular wave component could be absorbed. But for wave hitting right and left boundary in the first layer, propagation direction of wave is perpendicular to boundary and thus all wave would be absorbed.

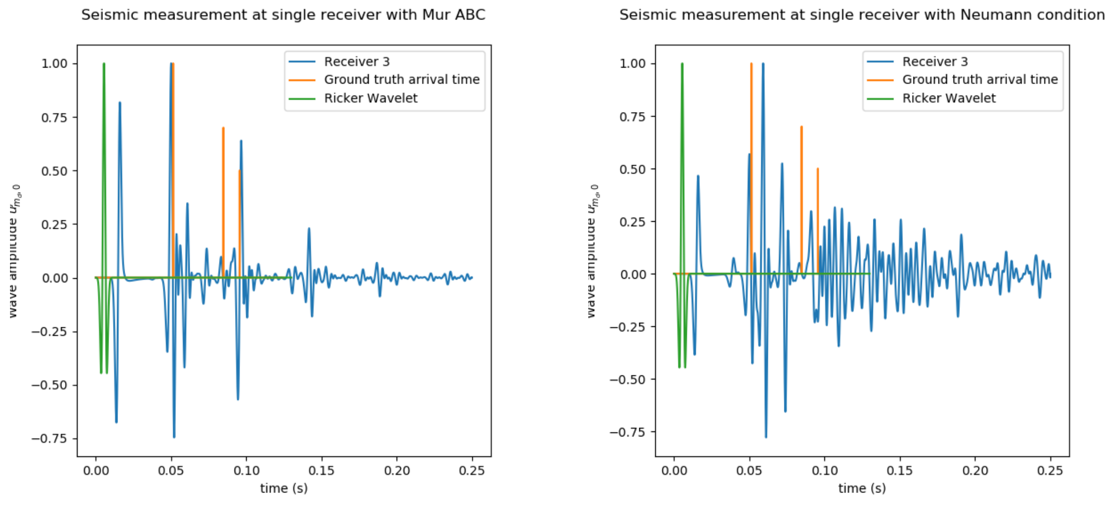
2.5.4 Receiver measurement analysis

The receiver measurement wave vector at receiver d is calculated with Mur ABC for ground cells given in (2.67):

$$\mathbf{a}^{j+1}_{m_d,0} = \begin{bmatrix} a^{j+1}_{m_d,0} \\ a^{j+1}_{m_d,0} \end{bmatrix} = \mathbf{a}^{j}_{m_d,0} + p_d \mathbf{a}^{j+1}_{m_d+1,0} - p_d \mathbf{a}^{j}_{m_d,0} \tag{2.67}$$

With $p_d = \frac{c_{m_d,0}\delta t - \delta x}{c_{m_d,0}\delta t + \delta x}$. As receiver measurement wave vector contains same wave amplitude component $a^j_{m_d,0}$ at jth time instant due to same source wave components, thus we only need to analyze one of it and then we summarize clean seismic measurement for dth receiver as $\mathbf{z}_d = [a^1_{m_d,0}, a^2_{m_d,0}, \dots, a^J_{m_d,0}]^T$. After FDTD implemenation with Ricker wavelet excitation, synthetic seismic measurement have already been generated. Generally speaking, there are two kinds of wave could be detected by receivers: **ground**

wave and **reflection wave**. The ground wave is usually first arrival observed in seismic measurement, as it travels horizontally on the ground, which has smaller propagating path, compared with reflection waves when the layer thickness is generally larger than source-to-receiver distance in reality. Firstly, the analysis of seismic measurement at receiver towards different boundary conditions on side boundaries of computational region is done to observe their influence to measurement.



(a) Mur ABC (b) Neuman condition

Figure 2.7: Seismic measurement with Mur ABC and Neumann condition

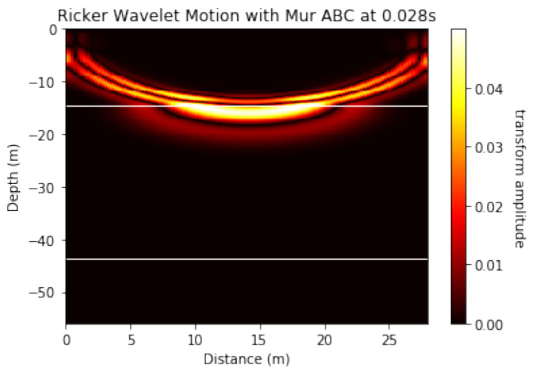
As wave diffusion makes its amplitude decay in spatial domain quickly, therefore we normalize the measurement to make it more comparable with source wavelet.

There are several points could be discovered from Fig 2.7:

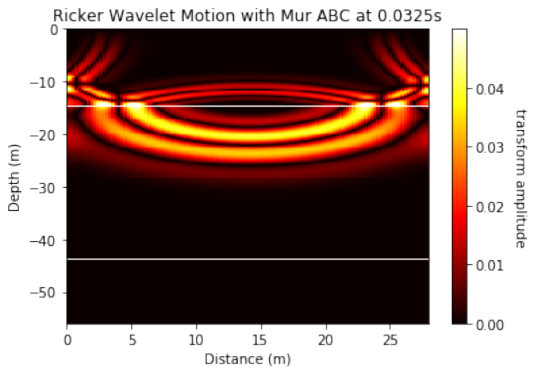
- the seismic measurement shows several seismic impulse responses which has similar envelope as source wavelet shown in green curve in Fig 2.7 (a). These seismic impulses reflect arrivals of waves detected at receiver.
- The first major peak indicates arrival of ground wave, followed by reflections from interfaces and bottom boundary. Additionally, measured wave amplitude shows an overall decaying trend along time, as wave diffusion means its amplitude is being smaller in spatial domain. Also,the ground surface and two sides are an absorbing boundary and bottom boundary is a reflective boundary. Thus the reflected and refracted wave during propagation and ultimately bounce back when it touches bottom surface and absorbed by side or ground boundaries.
- From Fig 2.7 (a), we observe there are still wave arrivals detected after reflection

of bottom boundary: main peaks at around 0.15s and 0.20s. This shows wave reverberation effect, as reflected wave is bouncing back and forth between interfaces at different layers and cause multiple reflection which can be seen from plot.

- From comparison of Fig 2.7 (a)-(b), superiority of Mur ABC for analysis is clearly shown as it absorbs unwanted side reflections and make measurement clean and easier to idenity arrival of waves. However, when we impose Neumann condition on side boundaries, incoming wave would be reflected back by boundary which can be seen in Fig 2.8. As a consequence, receivers would detect lots of unwanted reflection wave arrivals shown as many large peaks in subplot (b) of Fig 2.7 which makes whole measurement messed up . Additionally, measured waveform still show an overall decaying trend with Neumann condition on sides, because ground surface is set to be an absorbing boundary and thus it will continuously absorbing incoming wave.



(a) $t_j = 0.028$



(b) $t_j = 0.0325$

Figure 2.8: Wave reflection from side boundaries with Neumann condition

Also, we could capture normalized waveforms measured by different receivers with Mur ABC on both side boundaries which is shown in Fig 2.9. Due to space limit, we choose 6 among them shown in these 6 subplots correspond to 6 different receivers with increasing source-to-receiver distance on one side of source. The appearance of ground wave at different receiver show a delay trend, because receivers further away from source would detect it at later time point and also these impulse responses indicate ground wave follow a linear pattern, as wave propagation velocity at ground is assumed to be constant. Also peak amplitude corresponds ground wave is decreasing with further receiver, as wave is decaying and losing energy with propogation in spatial domain. Additionally, an interesting feature can be seen from Fig 2.9: major peak correponds to ground wave is less and less comparable with major peak correspond to first reflection at time around 0.05s. This can be explained as follows. Both reflection wave and ground wave decays in spatial domain. For ground wave, it is absorbed continuously by Mur ABC, while reflection

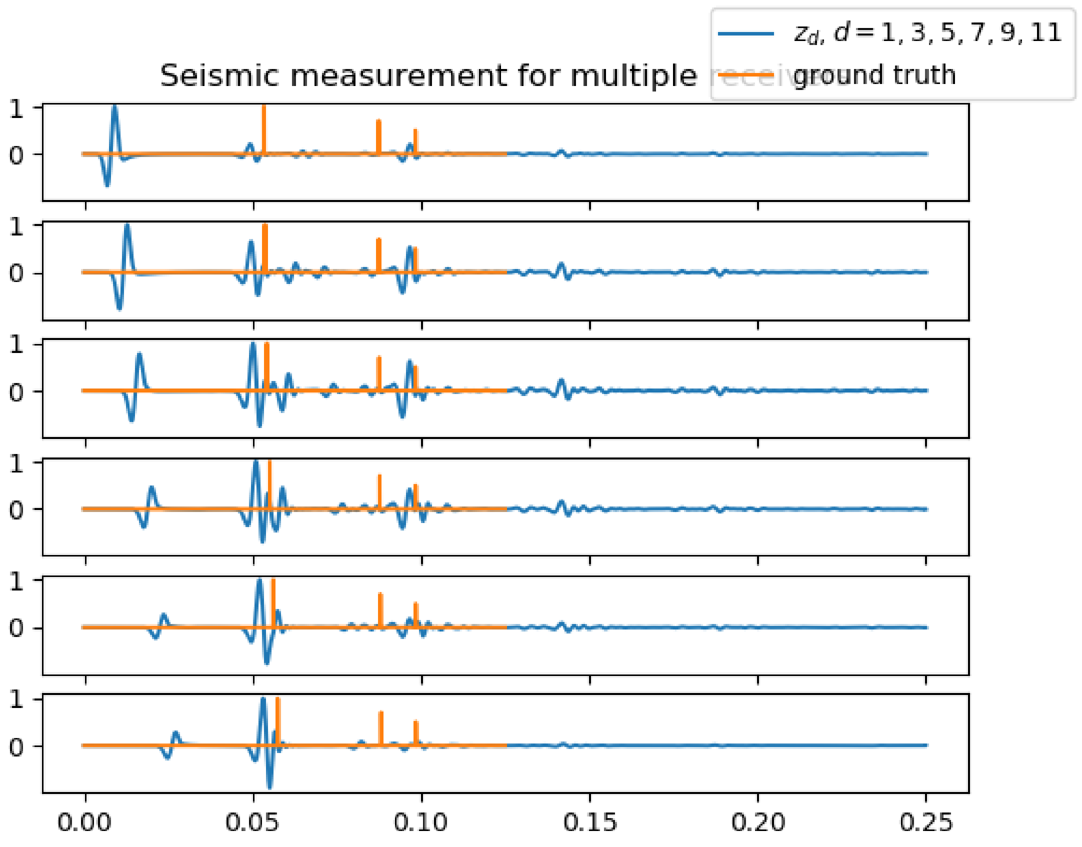


Figure 2.9: Seismic measurement at multiple receivers

wave from first interface loses energy due to interface reflection at first and wavefront expansion in the first layer and also the absorption of ground boundary. Therefore, reflection decays less rapid than ground wave with same physical propagating distance. For receiver very close to source, reflection from first interface loses more energy than ground wave, as ground wave doesn't being absorbed too much with small distance. But reflection wave need to expand in the whole layer and then being reflected by interface and absorbed by ground boundary, which would lose more energy.

However, reflection is becoming less weaker compare with ground wave for receivers further away. Then at one point, reflection wave would be equal than ground wave and then being stronger and stronger with further distance. This produces the general comparison pattern shown in Fig 2.9.

Chapter 3

Seismic measurement postprocessing

In the aforementioned data generation stage, the noiseless seismic measurement at multiple receivers have been generated to further analysis. However, in real data retrieving processing at receivers, measurement is always to be noisy due to ambient environment and sensor components which makes it is difficult to get useful information from them. Thus, in this chapter, we propose to do postprocessing regarding to noisy seismic measurement in terms of denoising with Wiener filter [6], deconvolution of noisy measurement and STA/LTA picking algorithm that help us to select a time point that corresponds to arrival of reflection wave from interfaces. Then we could proceed with Normal-moveout illustrated in the next chapter to estimate layer velocity and depth based on our picking time.

3.1 Ground truth arrival time of reflection wave

The ground truth of reflection wave arrival time from different layers at different receiver are quite important to compare picking travel time with them and get picking precision. Thus we back to the layer velocity and depth parameters set for simulation and calculate reflection time based on Snell's law: reflected ray refracted into deeper layer with a refraction angle depends on velocity ratio of two layers and incident angle of last layer. This scenario is clearly seen in Fig 3.1 with three curves with different colors. which we can know the ground truth travel time of different reflected wave reflected by interface f received by same dth receiver $t^r_{d,f}$:

$$\begin{aligned} t^r_{d,f} &= \sum_{i=1}^{f} \frac{2h_i}{\sin\theta_i v_i} \\ &= \sum_{i=1}^{f} \frac{2h_i}{\sqrt{1-(\frac{v_i\cos\theta_1}{v_1})^2}v_i} \end{aligned} \tag{3.1}$$

where h_i and v_i are layer depth and velocity for ith layer. For reflection from first

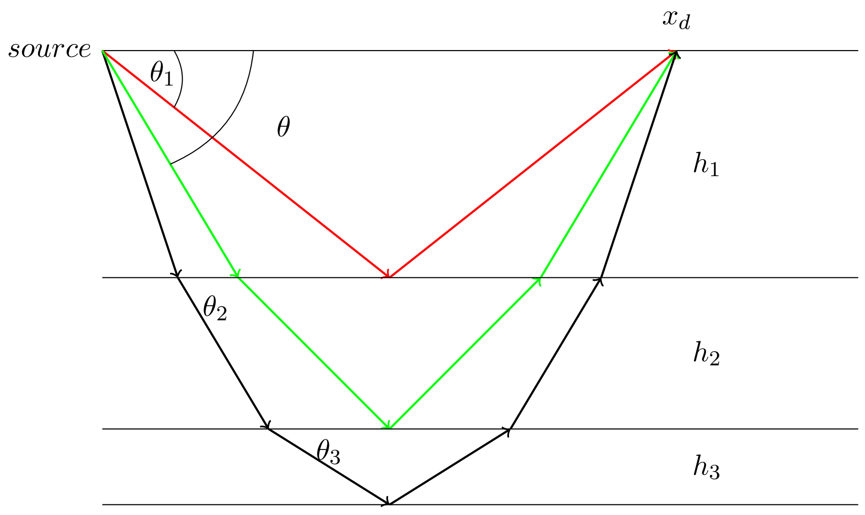


Figure 3.1: Reflection and refraction of seismic rays in multiple-layer model

interface, incidence angle at first layer θ_1 can be directly calculated as:

$$\theta_1 = \arctan\left(\frac{2h_1}{x_d}\right) \tag{3.2}$$

However for reflection from deeper layers for example green curve in Fig 3.1, we don't know the incidence angle at first layer θ but only know the source-receiver distance and the reflected wave from different layers will arrive back to same position, it is crucial to determine incidence angle correspond to this given receiver offset. The incidence angle is decreasing with deeper layers, as higher velocity in deeper layer. From formula (3.1) the only unknown is incidence angle for the first layer θ. With a given angle of incidence θ, the x-coordinate of reflected wave from layer f backs to ground can be calculated as following summation:

$$\begin{aligned} x_f &= \sum_{i=1}^{N} \frac{2h_i}{\tan\theta_i} \\ &= \sum_{i=1}^{N} \frac{2h_i \frac{v_i \cos\theta}{v_1}}{\sqrt{1-(\frac{v_i \cos\theta_1}{v_1})^2}} \end{aligned} \tag{3.3}$$

Then, we can choose a angle of incidence and calculated offset x_f with formula (3.3) and then proceed with a Brute-force searching for different angles and repeat this process for different layer f. x_f indicates the position of reflected ray will hit back on the ground surface for layer f. Then the problem turns to be finding a angle that make these calculated offset almost same with each other for each receiver. Therefore, for each layer at each receiver we test for different angles θ with angle test range divided into 2e5 evenly spaced angle values that $\theta \in (\theta_1, \frac{\pi}{2})$. For each test angle value we calculate (3.3) to test if it makes all calculated offset is very close to ground truth offset in an acceptable error range e:

$$|x_f - x_d| < e \quad \forall f \subset [1, \dots, f_r] \tag{3.4}$$

Once we find a valid text angle fulfill (3.4), we store it as generated ground truth arrival time for layer f at dth receiver. where x_d is the true source-to-receiver distance for dth receiver we know perfectly. we set $e = 0.001$ for simulation. Finally, we would get approximated ground truth reflection time by repeating above procedure for each layer at each receiver: $\mathbf{t}^r = [t^r_{1,1}, t^r_{1,2}, \dots, t^r_{D,f_r}]^T$

3.2 Denoising with Wiener filter

In reality, the sensor and ambient environment will cause some additive noise which would contaminate our measurement. In spectrum estimation domain, a typical denoising approach is to use wiener filter that estimate clean signal from noisy observed signal. The precondition of applying Wiener filter is we have knowledge of noiseless signal which we want to approximate.

To formulate this denoising technique, clean seismic measurement at dth receiver is a discrete data sequence $\mathbf{z}_d = [x_d[1], \dots, x_d[J]]^T$. Then we can write system model of seismic measurement containing noise as:

$$z^n_d[j] = z_d[j] + n[j] \tag{3.5}$$

where $z^n_d[j]$ is noisy observed seismic measurement at jth time intant, $z_d[j]$ is noiseless observed seismic measurement and $n[j]$ is addtive white Gaussian noise (AWGN) added in the system account for various noise sources at jth time instant.

Our goal is statistically estimate $z_d[j]$ from measured noisy signal $z^n_d[j]$. In order to remove noise from noisy seismic measurement, a wiener filter is applied for noisy seismic measurement. Firstly, the jth output of filtered seismic measurement is given by:

$$y[j] = \sum_{i=0}^{J} \kappa[i] z^n_d[J - i] \tag{3.6}$$

where $\boldsymbol{\kappa} = [\kappa[1], \kappa[2], \dots, \kappa[J]]^T$ is wiener coefficient vector. The Wiener Filter is designed to minimize minimum mean-square error (MMSE) between desired noiseless seimic measurement and noisy observed measurement, which is defined as expectation of squared error in (3.7) [6].

$$\begin{aligned} e =& \mathcal{E}(y[j] - z_d[j])^2 \\ =& \mathcal{E}(\sum_{i=0}^{J} \kappa[i] z^n_d[j-i])^2 + \mathcal{E}(z_d[j]^2) - 2\mathcal{E}(\sum_{i=0}^{J} \kappa[i] z^n_d[j-i] z_d[j]) \end{aligned} \tag{3.7}$$

taking the derivative of MMSE with respect to wiener coefficient and enforce it to zero, we can obtain:

$$\begin{aligned}\frac{\partial e}{\partial \kappa_i} =&2\mathcal{E}(\sum_{i=0}^{J}\kappa[i]z_d^n[j-i]z_d^n[j-i]) - 2\mathcal{E}(z_d^n[j-i]z_d^n[j])\\ =&2\sum_{i=0}^{J}\mathcal{E}[z_d^n[j-i]z_d^n[j-i]]\kappa_i - 2\mathcal{E}(z_d^n[j-i]z_d[j])\end{aligned} \tag{3.8}$$

we can do following substitution for autocorrelation and cross-correlation as:

$$\tau_x[i] = \mathcal{E}[z_d^n[j]z_d[j+i]] \tag{3.9}$$

$$\tau_{xs}[i] = \mathcal{E}[z_d^n[j]z_d[j+i]] \tag{3.10}$$

then we get:

$$\sum_{i=0}^{J}\tau_x[j-i]\kappa_j = \tau_{xs}[i] \tag{3.11}$$

finally we can solve the wiener coefficient vector from:

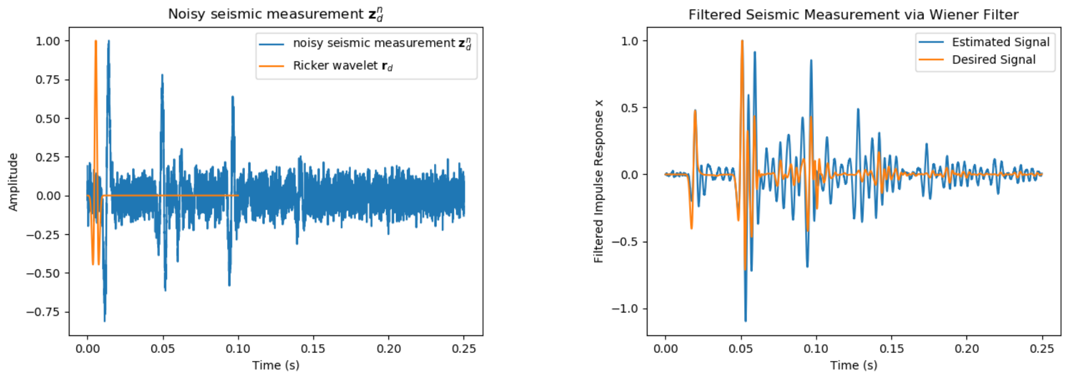
$$\underbrace{\begin{bmatrix}\tau_x[0] & \dots & \tau_x[J]\\ \dots & \dots & \dots\\ \tau_x[J] & \dots & \tau_x[0]\end{bmatrix}}_{\boldsymbol{\tau}}\underbrace{\begin{bmatrix}\kappa_0\\ \dots\\ \kappa_J\end{bmatrix}}_{\boldsymbol{\kappa}} = \underbrace{\begin{bmatrix}\tau_{xs}[0]\\ \dots\\ \tau_{xs}[J]\end{bmatrix}}_{\boldsymbol{\tau}_s} \tag{3.12}$$

then wiener coefficient vector $\boldsymbol{\kappa}$ is solved as with invertible matrix $\boldsymbol{\tau}^T\boldsymbol{\tau}$:

$$\boldsymbol{\kappa} = (\boldsymbol{\tau}^T\boldsymbol{\tau})^{-1}\boldsymbol{\tau}^T\boldsymbol{\tau}_s \tag{3.13}$$

From (3.13), the optimal wiener filter with corresponding coefficients is calculated with autocorreleation of noisy seismic measurement $\boldsymbol{\tau}$ and cross correlation between noisy and clean seismic measurement $\boldsymbol{\tau}_s$. For simulation purpose, we assume we have perfect knowledge about noiselss seismic measurement. To analyze wiener filter performance, noisy seismic measurement with SNR of 5dB is generated for simulation and final filtered results is shown in orange curve of Fig 3.2 (b) and we normalize filtered signal in order to make it comparable with desired signal.

The additive noise effect has been suppressed and Wiener filter recovers general pattern of desired noiseless seismic measurement by finding optimal filter coefficients that minimize MMSE, which can be seen by comparison of subplots (a)-(b) in Fig 3.2.



(a) Noisy seismic measurement (b) Denoising results from Wiener filter

Figure 3.2: Comparison of filtered and noisy seismic measurement

3.3 Deconvolution of Seismic measurement

Deconvolution is an essential step in seismic data processing, as it could suppress the effect of reverberations and prominent arrival events of seismic wave at receivers [2]. Typically, the seismic measurement could be formulated as a convolution model which describes noisy seismic trace $\mathbf{z}_d^n$ is a discrete convolution between source wavelet $\mathbf{R}_d$ and earth reflectivity series $\boldsymbol{\mu}_d$ at dth receiver.

$$\mathbf{z}_d^n = \mathbf{r}_d * \boldsymbol{\mu}_d + \mathbf{n}_d \tag{3.14}$$

Where wavelet $\mathbf{r}_d \in \mathbb{R}^{l_w}$ with length of l_w, $\boldsymbol{\mu}_d \in \mathbb{R}^{l_h}$, $\mathbf{z}_d \in \mathbb{R}^{l_w+l_h-1}$ and $\mathbf{n}_d \sim \mathcal{N}(0, \sigma^2)$ and has dimension $\mathbf{n}_d \in \mathbb{R}^{l_w+l_h-1}$. As measurement taken at receiver is impulse response from injecting Ricker wavelet. In the optimal case, injection of source signal with unit-amplitude Dirac impulse would allow us to retrieve optimal earth reflectivity profile that contains primary and multiple reflections from interfaces with recorded measurement shown as a series of vertical lines at receiver. Unfortunately, it is only feasible to produce wavelet impulse with finite duration, instead of a unit Dirac impulse with physical experiment. Thus, strictly speaking, convolution model of seismic measurment used here is a good approximation of optimal earth reflectivity response. Then we apply deconvolution to inversely estimate $\boldsymbol{\mu}_d$ from measured $\mathbf{z}_d^n$ with regularized least-square approach. Alternatively, convolution model in (3.14) could also be written in form of convolution matrix as:

$$\mathbf{z}_d^n = \mathbf{R}_d \boldsymbol{\mu}_d + \mathbf{n}_d \tag{3.15}$$

$\mathbf{R}_d$ is convolution matrix of wavelet and usually is a non-square matrix with dimension $\mathbf{R}_d \in \mathbb{R}^{(l_w+l_h-1)\times l_w}$and thereby we multiply with its transpose to make it square and invertible if $\mathbf{W}^T\mathbf{W}$ is well-conditioned. The goal of regularized least-square is to find estimated reflectivity series that minimizes following argument:

$$\underset{\mathbf{h}_d}{\operatorname{argmin}}\, \mathbf{b} = \frac{1}{\sigma^2}||\mathbf{z}_d^n - \mathbf{R}_d\hat{\boldsymbol{\mu}}_d||^2 + \lambda||\hat{\boldsymbol{\mu}}_d||^2 \tag{3.16}$$

The second term on the right-hand side of (3.16) is a Tikhonov regularization term applied to avoid ill-conditioned problem and overfitting. Simply calculate the derivative of this bias term $\mathbf{b}$ as:

$$\frac{\partial \mathbf{b}}{\partial \hat{\boldsymbol{\mu}}_{\boldsymbol{d}}} = \frac{1}{\sigma^2}\frac{\partial((\mathbf{z}_d^n)^2 - 2\mathbf{z}_d^n\mathbf{R}_d\hat{\boldsymbol{\mu}}_d + (\mathbf{R}_d\hat{\boldsymbol{\mu}}_d)^2)}{\partial \hat{\boldsymbol{\mu}}_{\boldsymbol{d}}} + 2\lambda\hat{\boldsymbol{\mu}}_d \tag{3.17}$$

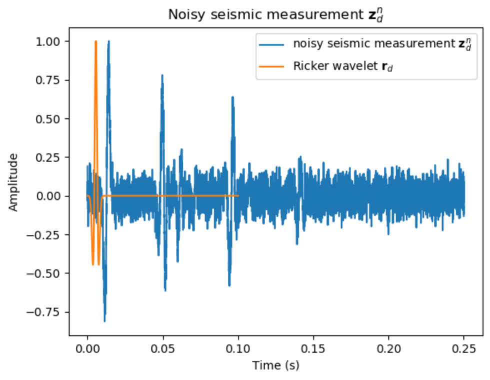
Proceed by setting this derivative to zero, we obtain:

$$\frac{1}{\sigma^2}(-2\mathbf{R}_d^T\mathbf{z}_d^n + 2\mathbf{R}_d^T\mathbf{R}_d\hat{\boldsymbol{\mu}}_d) + 2\lambda\hat{\boldsymbol{\mu}}_d = 0 \tag{3.18}$$

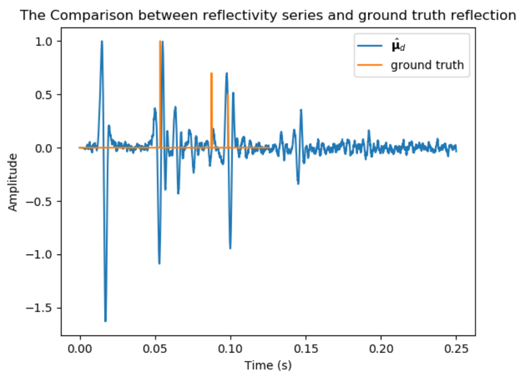
Rearrange terms in (3.18) we finally reach:

$$\hat{\boldsymbol{\mu}}_d = (\frac{\mathbf{R}_d^T\mathbf{R}_d}{\sigma^2} + \lambda\mathbf{I})^{-1}\frac{\mathbf{R}_d^T\mathbf{z}_d^n}{\sigma^2} \tag{3.19}$$

Assume noise variance of AWGN in noisy seimic measurement is perfectly known and wavelet for deconvolution is injected Ricker wavelet in FDTD with perfect knowledge for simulation. Then we apply this deconvolution processing to single noisy receiver measurement for SNR of 5dB with a regularization coefficient $\lambda = \frac{1}{\sigma^3}$ shown in Fig 3.3 (a), one point needs to be noted as we shift the reflectivity series by l_{peak} samples as source wavelet impulse is shifted by same amount and therefore its peak is situated at l_{peak}th time instant. Then we could have better correspondence of arrivals of reflection shown in reflectivity series. From comparison between Fig 3.3 (a)-(b), it is apparently shown that noise has been suppressed greatly and reflectivity series has been recovered from estimation of regularized least-square approach, which makes it is much easier to select time points correspond to picking travel time of reflection wave at receiver. Also, we still observe presence of additive noise in the deconvovled measurement. This deconvolution process need to be applied for each receiver measurement for further travel time picking. The deconvolved results for multiple noisy measurement with SNR of 5dB is shown in Fig 3.3 that estimated reflectivities are recovered from noisy measurement which ease the burden on further travel time picking.



(a) Noisy seismic measurement



(b) Estimated reflectivity from deconvolution

Figure 3.3: Comparison of deconvovled measurement and noisy seismic measurement

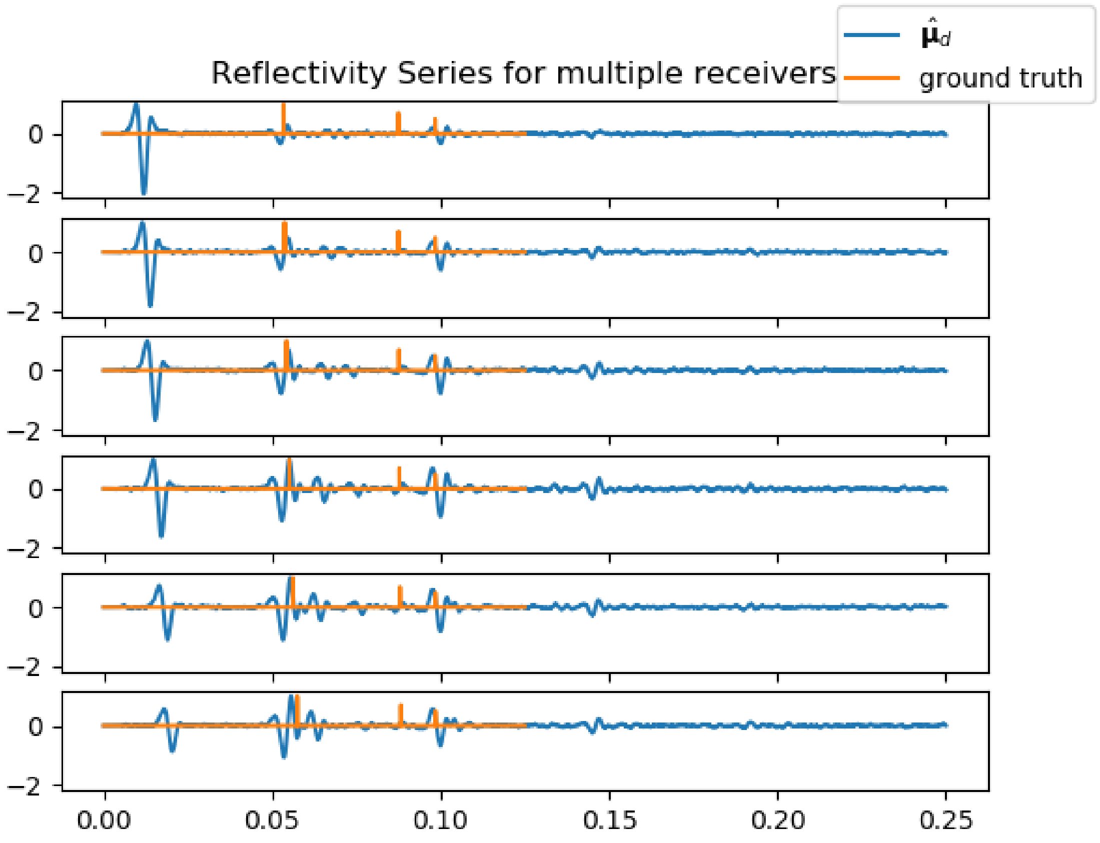


Figure 3.4: Reflectivity series $\boldsymbol{\mu}_d$ at multiple receivers

3.4 Travel time picking

In the aforementioned deconvolved stage, estimated reflectivity series are served as the input of picking travel time algorithm based on deconvolved seismic measurement. In this section, we illustrate two approaches to pick travel time of reflection wave, one based on deconvolved measurement and other one directly process noisy seismic measurement with STA/LTA picker.

3.4.1 Picking with deconvolved measurement

The reflectivity series contains primary reflection, which reflects only once at interface, multiple reflections and artificial reflection due to imperfection of Mur ABC. However, we only interest in primary reflection for further Normal-moveout implementation. A commonly used method to distinguish primary reflection is generally it has largest amplitude compared with multiple reflections, as multiple reflection happens when reflected wave reach upper interface from lower interface and then being reflected back to lower interface and thus it experience multiple enery losing process and show smaller amplitude compared with primary reflection. For reflection from side boundary, as only small portion of incoming wave would be reflected and it also has smaller amplitude. Therefore, time point associated with primary reflection should be around a local maximum peak of time window that we think there is primary reflection happens. In order to make selection process easier, ground truth arrival time of reflection wave is of assistance to get a sufficiently precise picking in this thesis. The whole picking algorithm based on reflectivity series is summarized in Algorithm 1.
The algorithm is explained as follows: line 1-3 represents a deconvolution stage of noisy seismic measurement to get corresponding reflectivity series $\hat{\boldsymbol{\mu}}_d$ at dth receiver. Line 5-10 follows with a selection of peaks and then choose local maximas from them. A peak is identified only if it is larger than right and left neighbor data points and its value is at least be threshold t_r in reflectivity series. The threshold is set to be 0 for simulation. Then we proceed to select closest local maxima with respect to ground truth of reflection wave travel time for total f_r layers in line 11. Line 12-18 performs a correction stage to check whether chosen local maximas are sufficiently close to ground truth. If it is too far away with a deviation larger than threshold T, then we choose data point as picking travel time $t_{d,f}$ in the reflectivity series cloeset to ground truth to guarantee a sufficiently precise picking for dth receiver at fth layer. On the contray, we would choose corresponding local maximum peaks as picking travel time if their deviations are within threhold. Threshold is set as $T = 5\delta t$ with δt for time resolution in FDTD. The picking result for first layer and comparison with ground truth is shown in Fig 3.5, where picking travel time are very close to ground truth. There is still deviation between picking and ground truth and rather small, even though it seems picking time covered ground truth in the Figure.

Algorithm 1 travel-time picking algorithm from deconvolved measurement

```
1:  for receiver $d = 1, 2, \dots, D$: do
2:      Taking noisy seismic measurement $\mathbf{z}_d^n$ with length $l$ and layer number $f_r$
3:      perform deconvolution
4:      $\hat{\boldsymbol{\mu}}_d = (\frac{\mathbf{R}_d^T \mathbf{R}_d}{\sigma^2} + \lambda \mathbf{I})^{-1} \frac{\mathbf{R}_d^T \mathbf{z}_d^n}{\sigma^2}$
5:      distinguish and select all peaks with threshold $t_r$ in $\boldsymbol{\mu}_d$
6:      Stack peaks: $P = [p_1, \dots, p_{max}]^T$
7:      distinguish local maximum among all peaks
8:      for $i = 2, \dots, max - 1$: do
9:          if $p_{i-1} < p_i < p_{i+1}$: then
10:             select $M = p_i$
            End if
        End for
11:     choose local maximum peaks $M_f$ closest to ground truth $t_f^r$ for layer $f$
12:     Correction picking stage
13:     for $f = 1, \dots, f_r$: do
14:         if $|M_f - t_f^r| <= T$: then
15:             $t_{d,f} = M_f$
16:         else
17:             choose data point which is closest to ground truth $x_d[c]$
18:             $t_{d,f} = x_d[c]$
            End if
        End for:$\rightarrow$t$_d = [t_{d,1}, \dots, t_{d,f_r}]^T$
```

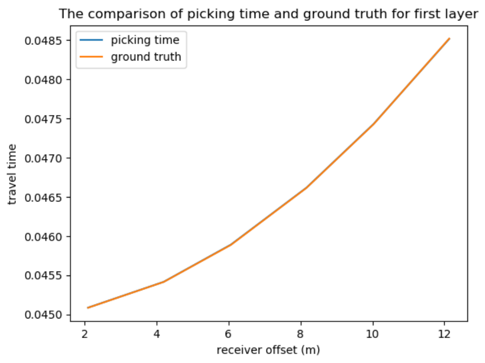


Figure 3.5: Picking travel time and ground truth

3.4.2 STA-LTA picking

In this section we propose a STA/LTA picker to pick arrival time of reflection wave from interfaces between layers and bottom boundary of simulation region to get measured

travel time of reflected wave [5]. Differing from previous picking method which heavily relies on deconvolved seismic measurement, We directly process noisy seismic measurement with STA/LTA method. This solves the problem of estimating noise variance of noisy measurement which is necessary to perform regularized least square for getting estimated deconvolved measurement from noisy measurement.
The idea of STA/LTA method starts with design of two windows: short-term average window, and long-term average window. For a noisy seismic measurement $\mathbf{z}_d^n$ at receiver d with total l data points, we first calculate average of absolute value for whole seismic measurement, this would reflect the energy of seismic measurement profile. Meanwhile, a short-term window is designed with specific window size w_r that moves along the seismic measurement profile from initial time instant and then the local average of the absolute of seismic measurement within this local window is calculated at each step. Then we divide this local average by long-term average as STA/LTA ratio which essentially describes an energy ratio shown as:

$$r_{STA}^i = \frac{\frac{1}{w_r}\sum_i^{i+w_r}|z_d^n[i]|}{\frac{1}{l}\sum_{i=1}^{l}|z_d^n[i]|} = \frac{short-term-absolute-average}{long-term-absolute-average} \tag{3.20}$$

We record this ration progressively along the seismic trace profile. When wave arrives at receiver, it will cause abrupt change of amplitude in the measured signal and therefore we could observe an impulse response in the STA/LTA ratio variation. One point needed to be noted is windowsize cannot be chosen too large. If it is too large, the moving window will may cover two consecutive arrival of waves and then we will only observe abrupt change of STA/LTA ratio for once and cause missing of wave detection. The simulation set up is given in table 3.1:

window size	0.002s
time resolution	5e-5s

Table 3.1: Simulation Setup for STA/LTA picker

As synthetic seismic measurement is generated with simulating wave propagation with FDTD and incorporate Mur absorbing boundary condition in the artifical boundaries to reduce unwanted reflection at sides. This is imperfect in 2D wave and there are still some waves will be reflected at the interfaces between layer boundaries. Therefore, we would observe some unwanted impulse response in the STA/LTA ratio profile. This can be seen by two abrupt impulse response between ground truth vertical lines at about 0.05s and 0.08s. The pseudo-code for STA/LTA picking is given in Algorithm 2 which is explained as follows: for each receiver, we take a noisy measurement and then let a short-term window moving across measurement series and record local absolute average at each time point to get STA/LTA ratio profile and normalize it, corresponds to line 2-7 in algorithm 2. Then select peaks with certain threshold which is 0 in simulation for STA/LTA ratio profile and compare them with ground truth arrival time, finally we could choose peaks closest to ground truth among all selected peaks, shown as line 7-11. Fig 3.6 shows STA/LTA ratio curve for single receiver after simulation. There it is clearly shown 3 impulse response appearing at around 0.05s, 0.08s and 0.09s respectively, which corresponds to 3 primary reflections from interface. Interestingly, there are two relatively large impulse response between first and second reflection. They are identified as arrival of artificial reflections from side boundaries, firstly arrive at receiver once which is wave beneath primary reflection wave in top layer in Fig 3.7 (a).

Algorithm 2 STA/LTA picking algorithm

```
 1: for receiver $d = 1, 2, \dots, D$: do
 2:     Taking noisy seismic measurement $\mathbf{z}_d^n$ with length $l$
 3:     choose specific window size $w_r$
 4:     while $i + w_r <= l$: do
 5:         calculate STA/LTA ratio
 6:         $r_{STA}^i = \frac{\frac{1}{w_r}\sum_i^{i+w_r} |z_d^n[i]|}{\frac{1}{l}\sum_{i=1}^{l} |z_d^n[i]|}$
        End: obtain STA/LTA ratio profile $\mathbf{r}_{STA}$
 7:     Normalize STA/LTA ratio profile
 8:     for $\mathbf{r}_{STA}$ do:
 9:         use peak finder to select all peaks in $\mathbf{r}_{STA}$ with given threshold $t_r$
10:         Choose peaks which are closest to ground truth $\mathbf{t}_f^r$ as picking $t_{d,f}$
        End for
11:     Stack picked travel time: $\mathbf{t}_d = [t_{d,1}, \dots, t_{d,f_r}]^T$
    end for:$\rightarrow$ $\mathbf{t}_d$ =0
```

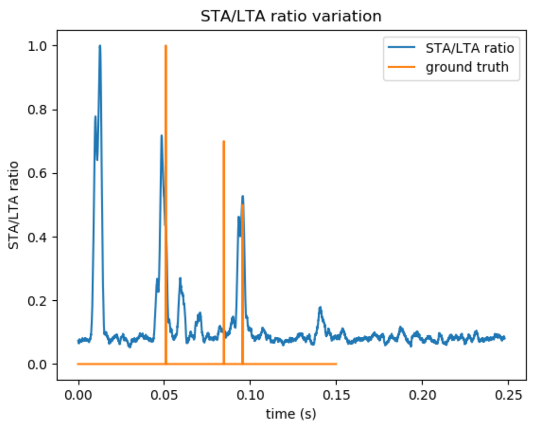
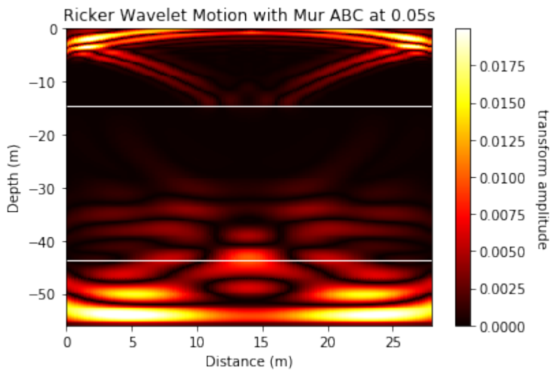
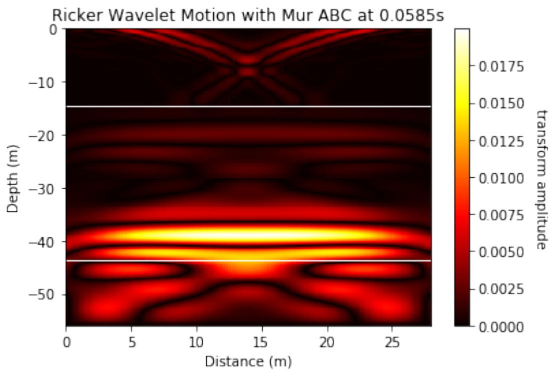


Figure 3.6: STA/LTA ratio profile at single receiver

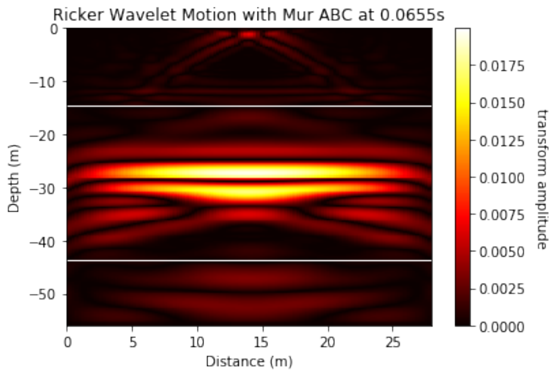
Then these two side reflections will interfere with each other shown in Fig 3.7 (b), as they propagate in opposite direction and cause second wavefront would be measured by receiver in Fig 3.7 (c). Fig 3.8 shows picking results of STA/LTA picker and picking from deconvovled profile and comparison with ground truth. Obviously, STA/LTA picker shows much worse performance than latter, as it doesn't have a correction stage in the picking algorithm from deconvoled profile. Certainly, we could also do this correction



(a) $t_j = 0.05$s



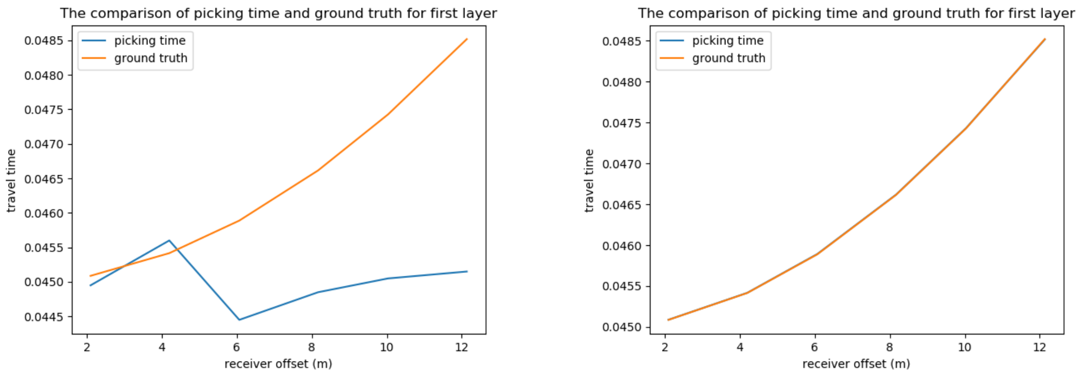
(b) $t_j = 0.0585$s



(c) $t_j = 0.0655$s

Figure 3.7: Side reflections and their interference

in STA/LTA picking, but interesting point is we want to test how Normal-moveout algorithm performs under an accurate and inaccurate picking in the next chapter and thus picking correction is not applied for STA/LTA picker.



(a) Picking from STA/LTA picker (b) picking from deconvolved measurement

Figure 3.8: Picking results comparison

Chapter 4

Normal-moveout (NMO) implementation

A frequently used technique of estimating subsurface structrue and wave velocity is Normal-moveout (NMO), which describes that increasing distance between source and receiver would cause additional delay time of arrival that detected at receiver and show a hyperbolic pattern [2]. In this chapter, centralized NMO and two distributed NMO algorithms are presented to show how we proceed aforementioned estimation tasks with picking travel time at different receivers for different layers and evaluation of numerical results from NMO algorithms.

4.1 Centralized NMO algorithm

4.1.1 Principle and motivation

The NMO algorithm scenario is shown in Fig 4.1, where two receiver arrays symmetrically positioned on both sides of source shown as small black bots and then measure reflection wave from interface.

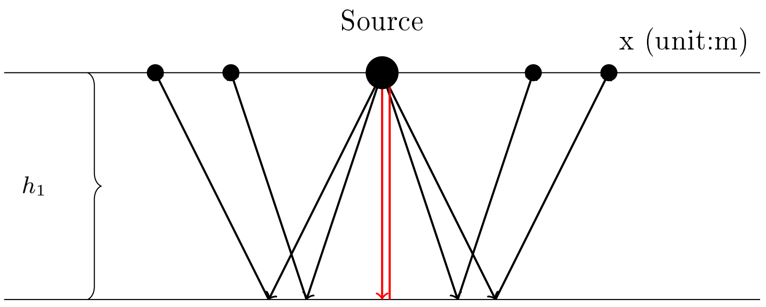


Figure 4.1: Normal-moveout Illustration

The two-way travel time (TWT) of reflection wave starts from source and then reflected by interface and arrives at receiver could be calculated with simple geometry with TWT at normal incidence angle $t_{0,1}$ with propagation velocity of first layer at dth receiver $v_{d,1}$ corresponds to optical path shown as red line in Fig 4.1 is given in (4.1):

$$\begin{aligned} t^r_{d,1} =& \frac{2}{v_{d,1}}(\sqrt{h_1^2 + (\frac{x_d}{2})^2}) \\ =& \frac{2h_1}{v_{d,1}}(\sqrt{1 + (\frac{x_d^2}{4h_1^2})}) \\ =& t_{0,1}(\sqrt{1 + (\frac{x_d^2}{4h_1^2})}) \end{aligned} \tag{4.1}$$

Where $t^r_{d,1}$ is ground truth arrival time of reflected ray at dth receiver for first layer. Then for dth receiver at first layer, we get:

$$t^r_{d,1} = t_{0,1}(\sqrt{1 + (\frac{x_d^2}{4h_1^2})}) \tag{4.2}$$

$t_{0,1}$ could be estimated by time measurement with nonlinear least-square fitting which is explained in the further section. As we have a series of measurement of reflection wave arrival time recorded at receiver array stacked as: $\mathbf{t} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{f_r}]^T$. A frequently used approxiamtion in geophysical exploration is by taking binomial expansion of expression under square root in (4.1), we could obtain:

$$1 + (\frac{x_d^2}{4h_1^2}) = 1 + \frac{1}{2}(\frac{x_d}{2h_1})^2 + o(n) \tag{4.3}$$

then we could neglect higher order terms for the case if source is close to the receivers and therefore reflected ray would not deviate from vertical angle too much, and also receiver offset is much small compared to layer depth. Finally we reach to the estimated propagation velocity with picking travel time $t_{d,n}$ in the first layer shown in (4.4):

$$\hat{v}_{d,1} \approx \sqrt{\frac{x_d^2}{2(t_{d,1} - t_{0,1})t_{0,1}}} \approx \sqrt{\frac{x_d^2}{2t_f t_{0,1}}} \tag{4.4}$$

where $t_f = t_{d,1} - t_{0,1}$ is time difference between two-way travel time of reflected ray at normal incidence angle and at receiver for first layer. However, for our simulation scenario source-to-receiver distance is not so small compared with layer thickness due to computational time consideration with FDTD for analysis. Therefore, we directly calculate estimated propagation velocity $\hat{v}_{d,1}$ from (4.1) as:

$$\hat{v}_{d,1} = \frac{x_d}{\sqrt{t_{d,f}^2 - t_{0,1}^2}} \tag{4.5}$$

Where $t_{d,1}$ is two-way travel time for first layer at dth receiver that is obtained from picking algorithm in the previous chapter. For multiple layers, it is still applicable (4.5) for the equation , if we consider total f homogenous layers is a single homogeneous layer but with another propagation velocity $v^r_{rms,f}$ that gives same travel time of reflection wave at from source to receivers. Then we may solve individual wave propagation velocity for each layer based on $v^r_{rms,f}$. $v^r_{rms,f}$ is called root-mean square velocity which is calculated as weighted average of internal layer velocity for all layers. In order to derive its mathematical expression, we firstly come to replace estimated root mean-square velocity $\hat{v}^{rms}_{d,f}$ with picking travel time into Normal moveout equation:

$$t^2_{d,f} = \hat{t}^2_{0,f} + \frac{x^2_d}{\hat{v}^{rms2}_{d,f}} \tag{4.6}$$

Where $t_{d,f}$ is picking two-way travel time for fth layer at dth receiver and $\hat{t}_{0,f}$ is estimated two-way vertical travel time for layer f. Now, let's assume angle between incidence of seismic ray and normal direction at fth layer as θ_f, then we could calculate distance between the position it backs to the ground surface for the fth layer and shot source for vertical ray as:

$$\begin{aligned} \sum_{f=1}^{n} x_f &= \sum_{f=1}^{f} 2\eta_f \hat{v}_{d,f} \tan\theta_f \\ &\approx \sum_{i=1}^{f} 2\eta_f \hat{v}_{d,f} \sin\theta_f \end{aligned} \tag{4.7}$$

η_f is one-way travel time of seismic ray in fth layer. The last line in (4.7) is done as seismic ray is almostly vertically injected from source and thus η_f is quite small. Meanwhile, for a particular layer, we could formulate a calculus with respect to θ_f as:

$$\sin\theta_f = \hat{v}_{d,f} \frac{dt_{d,f}}{dx_d} \tag{4.8}$$

Then, we differentiate normal moveout formula with respect to x_d as:

$$t_{d,f} \frac{dt_{d,f}}{dx_d} = \frac{x_d}{(\hat{v}^{rms}_{d,f})^2} \tag{4.9}$$

Subsequently we substitute $\frac{dt_{d,f}}{dx_d}$ into (4.10) and solve for $\sin\theta_f$:

$$\sin\theta_f = \frac{x_f \hat{v}_{d,f}}{t_{d,f}(\hat{v}^{rms}_{d,f})^2} \tag{4.10}$$

Finally we replace (4.10) into (4.7) and reach:

$$(\hat{v}^{rms}_{d,f})^2 \sum_{f=1}^{f} x_d \approx \sum_{f=1}^{f} x_d \frac{2\eta_f (\hat{v}_{d,f})^2}{t_{d,f}} \tag{4.11}$$

Where $t_{d,f} = 2\sum_{f=1}^{f}\eta_f$. Ultimately, we divide both sides of (4.11) by $\sum_{i=1}^{f}x_f$ and take limit $\sum_{f=1}^{f}x_f \to 0$ for the vertical ray and obtain expression of estimated root mean square velocity:

$$\hat{v}_{d,f}^{rms} = \sqrt{\frac{\sum_{f=1}^{f}\eta_f\hat{v}_{d,f}^2}{\sum_{f=1}^{f}\eta_f}} \tag{4.12}$$

Critical point of this formula is η_f must be one-way vertical travel time due to consideration of precision, otherwise this formula would not be accurate to calculate root-mean square velocity. Based on expression of root mean-square velocity in (4.12), internal velocity of fth layer v_f for dth receiver could be calculated with Dix formula [2]:

$$\hat{v}_{d,f} = \sqrt{\frac{(\hat{v}_{d,f}^{rms})^2\hat{t}_{0,f} - (\hat{v}_{f-1}^{rms})^2\hat{t}_{0,f-1}}{\hat{t}_{0,f} - \hat{t}_{0,f-1}}} \tag{4.13}$$

Where η_{f-1} and η_f is oneway travel time of reflected wave in layer $f-1$ and f. Then the thickness of each layer could be reconstructed with reconstructed layer velocity with assistance of estimated travel time that need to propgagte through fth layer at dth receiver $\hat{h}_{d,f}$ in (4.14):

$$\hat{h}_{d,f} = \frac{1}{2}(\hat{t}_{0,f} - \hat{t}_{0,f-1})\hat{v}_{d,f} \tag{4.14}$$

Where $\hat{t}_{0,f} - \hat{t}_{0,f-1}$ is time that seismic ray need to spend to travel for layer f twice with vertical incidence angle and for ground surface we have $\hat{t}_{0,0} = 0$ Before we implement normal moveout, the picking arriavl time of reflection waves from interfaces are obtained from processing of multiple receiver measurement with previously mentioned picking algorithm 1 from deconvolved profile. They form a picking time data matrix that we can use to estimate t_0.

The arrival time from multiple sensors form 3 curves with a general hyperbolic trend but not perfect hyperbola shown in Fig 4.2, each one is associated from reflection of one layer. These picking time form basis of estimating two-way vertical travel time $t_{0,f}$ for each layer. For deeper layers, variation in picking time is smaller because larger wave velocity which can also be seen in Fig 4.2.

Levenberg-Marquardt algorithm for nonlinear least-square fitting

Estimated $t_{0,f}$ is significant to solve estimated layer velocity from Normal-moveout implementation in the previous analysis.

As NMO equation in (4.6) is a nonlinear equation and then we could write estimated picking travel time from a nonlinear model with associated parameter $\mathbf{p}_f = [\hat{u}_f, \hat{t}_{0,f}]^T$

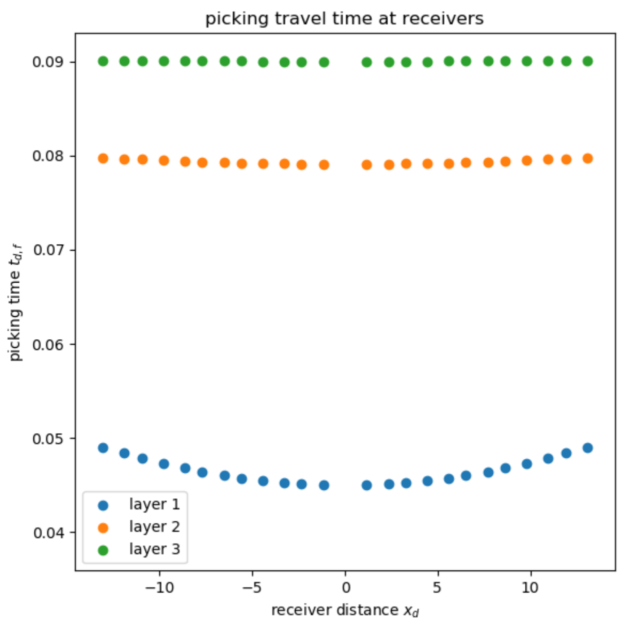


Figure 4.2: picking travel time from deconvolved measurement

and receiver distance x_d for layer f in (4.15) and also corresponding ground truth model in (4.16):

$$\hat{t}_{d,f}^2 = t(x_d, \mathbf{p}_f) = \hat{u}_f x_d^2 + \hat{t}_{0,f}^2 \tag{4.15}$$

$${t_{d,f}^r}^2 = u_f x_d^2 + t_{0,f}^2 \tag{4.16}$$

Where $\hat{u}_f = \frac{1}{\hat{v}_{rms,f}^r}$ is slowness which is essentially inverse of estimated root mean-square velocity square. Then we could proceed by defining a sum of square error term between picking travel time and estimated travel time for each layer:

$$\begin{aligned} S_f(\mathbf{p}_f) &= \sum_{d=1}^{D} (t_{d,f}^2 - \hat{t}_{d,f}^2)^2 \\ &= ||\mathbf{t}_f^2 - \hat{\mathbf{t}}_f^2||^2 \end{aligned} \tag{4.17}$$

The goal is to find optimal parameters $\hat{u}_f$ and $\hat{t}_{0,f}$ minimize above sum of square error. In real simulation, a standard python package *Scipy.optimize.curve$\underline{f}$it* is applied to manage it with Levenberg-Marquardt algorithm which has general structure as follows [7]. For detailed implementation, it can be found in [1]. The Levenberg-Marquardt algoithm originates from Gauss-Newton algorithm which has typical form starts with an initial guess of parameters that replaces update parameter at next time step with new parameters:

$$\mathbf{p}_f^{k+1} = \mathbf{p}_f^k + \Delta\mathbf{p} \tag{4.18}$$

Where $\Delta\mathbf{p}$ is step size in parameter update procedure. The first order approximation of synthesized picking time is:

$$t(x_d, \mathbf{p}_f^{k+1}) = t(x_d, \mathbf{p}_f^k) + \frac{\partial t(x_d, \mathbf{p}_f^k)}{\partial \mathbf{p}_f^k}\Delta\mathbf{p} \tag{4.19}$$

Where $\mathbf{J}_d = \frac{\partial t(x_d, \mathbf{p}_f^k)}{\partial \mathbf{p}_f^k}$ is gradient of NMO model $t(x_d, \mathbf{p}_f^k)$ with respect to parameters $\mathbf{p}_f^k$. The updated parameters should minimize sum of square error and therefore we could set its derivative with respect to step size to zero:

$$\frac{\partial ||\mathbf{t}_f - t(x_d, \mathbf{p}_f^k) - \mathbf{J}\Delta\mathbf{p}||^2}{\partial \delta} = 0 \tag{4.20}$$

With $\mathbf{J} = [\mathbf{J}_1, \mathbf{J}_2, \ldots, \mathbf{J}_D]^T$ is Jacobian matrix. Finally we would get:

$$\mathbf{J}^T\mathbf{J}\Delta\mathbf{p} = \mathbf{J}^T(\mathbf{t}_f - t(x_d, \mathbf{p}_f^k)) \tag{4.21}$$

(4.21) represents a classic Gauss-Newton update, however in Levenberg Marquardt algorithm, we add an additional damping factor γ into right-hand side of (4.21) to adaptively modify step size in parameter update $\Delta\mathbf{p}$. Then we could solve for step size at each iteration as:

$$\Delta\mathbf{p} = (\mathbf{J}^T\mathbf{J} + \gamma\mathbf{I})^{-1}\mathbf{J}^T(\mathbf{t}_f - t(x_d, \mathbf{p}_f^k)) \tag{4.22}$$

A nonlinear-least square fitting algorithm to estimate parameters for each layer is formulated in Algorithm 3 with pseudocode and is explained as follows: Firstly, we extract picking travel time at receivers for single layer and make an initial guess of parameters corresponds to line 1-3, then put initial guess parameters into inner loop of algorithm update with sufficient large time steps k_{max}. At each step, we calculate sum of square error, shown in line 6 and proceed with parameter shift with chosen initial damping factor γ in line 7-8. Afterwards, a new sum of square error is calculated with based on updated parameter in line 8. To adaptively modify step size, we firstly check if current sum of square error is smaller than required fitting error S_{min}. If it is, we break and if not we compare new sum of square error with previous one and increase γ if it is larger, or decrease it if it is smaller. This corresponds to line 12-17. This adaptive operation accelerates algorithm convergence when updated parameters reduce sum of square error and slow it when updated parameters are worse which increases sum of square error. Finally we would get an optimal parameters that minimize sum of sqaure error.

Centralized NMO algorithm

Based on previous NMO analysis, we can formulate a centralized NMO algorithm that each receiver has access to picking travel time from all other receivers and they can transmit data to a central receiver. Then each receiver could perform nonlinear least-square

Algorithm 3 Levenberg Marquardt algorithm

```
1:  for layer $f = 1, 2, \dots, f_r$: do
2:      Taking picking travel time for reflection wave $\mathbf{t}_f$
3:      Start with initial guess $\mathbf{p}_1 = [\hat{u}_f^1, \hat{t}_{0,f}^1]^T$
4:      for iteration $k = 1, 2, \dots, k_{max}$ do
5:          calculate sum of square error
6:          $S_f(\mathbf{p}_f) = ||\mathbf{t}_f^2 - t(x_d, \mathbf{p}_f^k)||^2$
7:          Update parameters
8:          $\mathbf{p}_f^{k+1} = \mathbf{p}_f^k + (\mathbf{J}^T\mathbf{J} + \gamma\mathbf{I})^{-1}\mathbf{J}^T(\mathbf{t}_f - t(x_d, \mathbf{p}_f^k))$
9:          recalculate sum of square error
10:         $S_f(\mathbf{p}_f + \Delta\mathbf{p}) = ||\mathbf{t}_f^2 - t(x_d, \mathbf{p}_f^{k+1})||^2$
11:         check and modify step size
12:         if $S_f(\mathbf{p}_f + \Delta\mathbf{p}) < S_{min}$ then
13:             break
            End If
14:         if $S_f(\mathbf{p}_f + \Delta\mathbf{p}) < S_f(\mathbf{p}_f)$ then
15:             increase $\gamma$
16:         else
17:             decrease $\gamma$
            End If
        End for
    End for
```

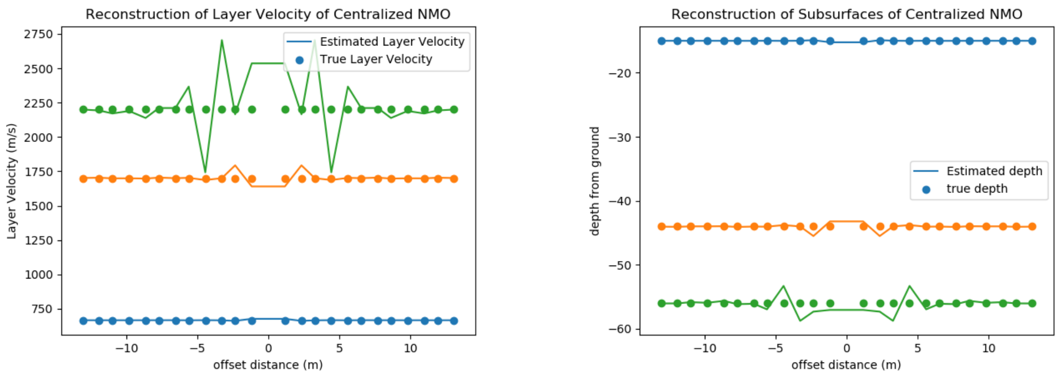
fitting to estimate parameters and then estimate layer velocity and depth in a centralized manner, as each receiver could be one central node. The algorithm structure is shown in Algorithm 4 and explained as follows: firstly, each receiver obtains picking travel time with STA/LTA picker or from deconvolved measurement corresponds to line 1-8. Then we loop over all receivers and each receiver is served as central node subsequently and gather picking travel time for each layer from all other receivers and then estimate parameter slowness and two-way vertical travel time at all receivers based on these picking time, corresponds to line 9-12. Then we proceed with a denoising stage to correct picking by recalculating picking travel time based on estimated parameters and receiver position x_d in line 13-14. This is critical for NMO, as small deviation of picking travel time will cause large estimation error in layer velocity. Afterwards, root mean square velocity is calculated based on NMO equation in line 15-16. Line 17-21 follows with a recursive calculation of internal wave propagagtion velocity in deeper layer based on calculated root mean square velocity for two consecutive layers. Additionally, for first layer calculated root mean square velocity is same as estimated wave propagation velocity shown in line 18. In line 22-23, estimated layer thickness $\hat{v}_{d,f}^c$ can be done with estimated two-way vertical travel time between two consecutive layers f and $f-1$ and estimated internal layer velocity $\hat{v}_{d,f}^c$. To the end of algorithm, we would obtain an estimated layer velocity and thickness at each receiver for each layer.

Algorithm 4 Centralized NMO algorithm

```
 1: for $\mathbf{z}_d^n$ d=1,...,D: do
 2:     choose picking method
 3:     if picking from deconvolved measurement then
 4:         $\hat{\boldsymbol{\mu}}_d = (\frac{\mathbf{W}^T\mathbf{W}}{\sigma^2} + \lambda\mathbf{I})^{-1}\frac{\mathbf{W}^T\mathbf{z}_d^n}{\sigma^2}, \lambda = \frac{1}{\sigma^3}$
 5:         travel-time picking
 6:     else
 7:         picking from STA/LTA picker
        end if
 8:     $\mathbf{t} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{f_r}]^T$
    end for
 9: for receiver $d = 1, 2, \ldots, D$: do
10:     gather picking travel time from all other receivers $\mathbf{t}$ at central receiver $d$
11:     for layer $f = 1, 2, \ldots, f_r$: do
12:         perform nonlinear least-square fitting $\mathbf{p} = [\hat{u}_f^c, \hat{t}_{0,f}^c]^T$
13:         recalculate picking travel time
14:         $\hat{t}_{d,f}^2 = \sqrt{\hat{u}_f x_d^2 + \hat{t}_{0,f}^2}$
15:         Solve for root mean square velocity
16:         $\hat{v}_{d,f}^{rms} = \frac{x_d}{\sqrt{\hat{t}_{d,f}^2 - \hat{t}_{0,f}^2}}$
17:         check if picking corresponds to first layer
18:         if f==1 then $\hat{v}_{d,1} = \hat{v}_{d,1}^{rms}$
19:             solve internal layer velocity recursively for multiple layer case
20:         else
21:             $\hat{v}_{d,f}^c = \sqrt{\frac{(\hat{v}_{d,f}^{rms})^2\hat{t}_{0,f} - (\hat{v}_{f-1}^{rms})^2\hat{t}_{0,f-1}}{\hat{t}_{0,f} - \hat{t}_{0,f-1}}}$
            End If
22:         Solve layer thickness
23:         $\hat{h}_{d,f}^c = \frac{1}{2}(\hat{t}_{0,f} - \hat{t}_{0,f-1})\hat{v}_{d,f}$
        End For:$\rightarrow \hat{\mathbf{v}}_d^c = [\hat{v}_{d,1}^c, \ldots, \hat{v}_{d,f_r}^c]^T, \hat{\mathbf{h}}_d^c = [\hat{h}_{d,1}^c, \ldots, \hat{h}_{d,f_r}^c]^T$
    End For: $\rightarrow \hat{\mathbf{v}}^c = [\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \ldots, \hat{\mathbf{v}}_D]^T, \hat{\mathbf{h}}^c = [\hat{\mathbf{h}}_1, \hat{\mathbf{h}}_2, \ldots, \hat{\mathbf{h}}_D]^T$
```

4.1.2 Estimation results

Firstly, we execute Algorithm 4 to implement Normal-moveout in a centralized fashion but without recalculating of picking in line 14 of Algoirthm 4 to investigate how the picking travel time deviation influences estimation performance. The estimation performance regarding to layer velocity and depth is shown in Fig 4.3 (a)-(b). Obviously, estimation performance is worse in the deeper layers, which is because we solve internal layer velocity based on calculated root-mean square velocity of current layer and the layer above it. Thus, it turns out the error in the picking travel time will influence velocity estimation result in the first layer and then indirectly influence velocity estimation result in second layer from recursive calculation in line 13 of Algorithm 4 and so forth, which concludes estimation performance is degrading with deeper layers. Meanwhile, we also observe the receivers which are close to source have worse estimation performance than those further away from source. This can be explained as follows. if we consider our picking $t_{d,f}$ is



(a) Velocity estimation (b) Depth estimation

Figure 4.3: Classic NMO estimation without recalculation of picking

deviated from ground truth $t^r_{d,f}$ with model as:

$$t^r_{d,f} = t_{d,f} + \Delta t_{d,f} \tag{4.23}$$

Then we rewrite root mean square velocity estimate formula as:

$$\begin{aligned}
\hat{v}_{d,f}^{rms} &= \frac{x_d}{\sqrt{(t_{d,f}^2 - \hat{t}_{0,f}^2)}} \\
&= \frac{1}{\sqrt{\frac{(t_{d,f}^2 - \hat{t}_{0,f}^2)}{x_d^2}}} \\
&= \frac{1}{\sqrt{\frac{(t_{d,f}^r - \Delta t_{d,f})^2 - \hat{t}_{0,f}^2}{x_d^2}}} \\
&= \frac{1}{\sqrt{\underbrace{\frac{({t_{d,f}^r}^2 - \hat{t}_{0,f}^2)}{x_d^2}}_{A} - \underbrace{\frac{2\Delta t_{d,f}\hat{t}_{0,f}}{x_d^2}}_{B} + \frac{\Delta t_{d,f}^2}{x_d^2}}} \\
&\approx \frac{1}{\sqrt{\underbrace{\frac{({t_{d,f}^r}^2 - \hat{t}_{0,f}^2)}{x_d^2}}_{A} - \underbrace{\frac{2\Delta t_{d,f}\hat{t}_{0,f}}{x_d^2}}_{B}}}
\end{aligned} \tag{4.24}$$

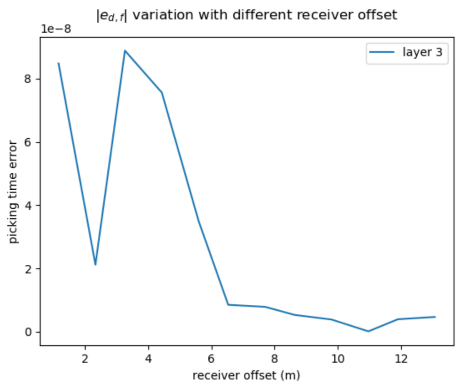
The approximation at the end line can be done because generally deviation of picking is very small compare with estimated t_0.

$$\Delta t_{d,f} << \hat{t}_{0,f} \tag{4.25}$$

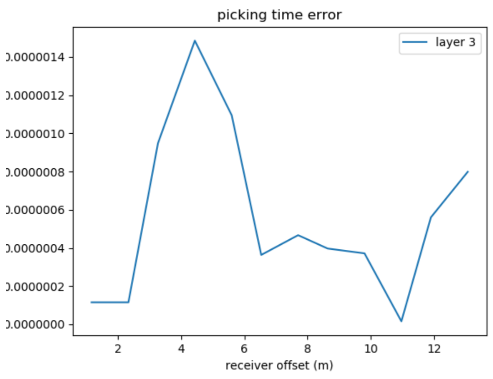
The term A associated ground truth arrival time and can be interpreted as optimum estimation with a given t_0 estimate. From term B we can find out how picking time influences the estimation performance. In optimum case, we want to vanish term B. In practice, the deviation of picking affects the estimation performance not directly, but scaled by receiver offset square x_d^2. As t_0 estimate is same for same layer in centralized normal moveout. Therefore we investigate deviation term $e_{d,f}$:

$$e_{d,f} = \frac{|\Delta t_{d,f}|}{x_d^2} \tag{4.26}$$

The variation of this term should reflect variation trend in estimation curve variation. To investigate it, we take picking time from third layer as an example and show it in Fig. 4.4 (a), we could find out $e_{d,f}$ show an oscillating pattern for receiver close to source within a distance or roughly 6 meters and decreases with increasing distance generally. This also reflects pattern in the estimation performance in Fig 4.3. This trend is because influence of picking time error is scaled by receiver offset square, which turns out deviation term



(a) variation of $e_{d,f}$



(b) picking time error $\Delta t_{d,f}$ unit:s

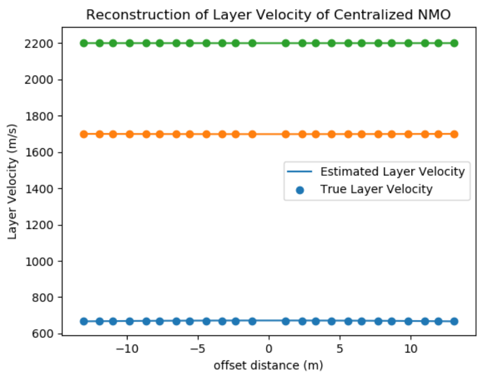
Figure 4.4: Classic NMO estimation estimation error pattern analysis

$e_{d,f}$ is generally decreasing after divided by relative dramatic increasing receiver distance, even though picking time deviation $\Delta t_{d,f}$ varies up and down shown in Fig 4.4 (b).

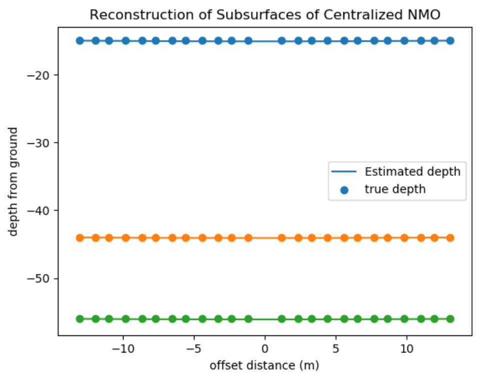
From above estimation performance of NMO, it is undoubtedly picking time si significant to estimation performance, then we add correction stage in line 6 in Algorithm 4 to observe how estimation is carried out with this modification shown in Fig 4.5. Clearly, estimated layer velocity and depth are almostly same and very close to ground truth which shows very good estimation results. This prominents the significance of repicking of travel time that makes new picking travel time form a perfect hyperbola that defined by estimated parameters from fitting. As a consequence, NMO will get same estimated results at each receiver. This can be stated as follows. Firstly, we also come to estimated layer velocity with repicking:

$$\begin{aligned}\hat{v}_{d,f}^{rms} &= \frac{x_d}{\sqrt{\hat{t}_{d,f}^2 - \hat{t}_{0,f}^2)}} \\ &= \frac{x_d}{\sqrt{\hat{u}_f x_d^2 + \hat{t}_{0,f}^2 - \hat{t}_{0,f}^2}} \\ &= \frac{1}{\sqrt{\hat{u}_f}}\end{aligned} \tag{4.27}$$

As each receiver would have all picking travel time for nonlinear least-square fitting and thus estimated parameters $\hat{u}_f$ and $\hat{t}_{0,f}$ in (4.27) is same for all receivers, which turns out root mean-square velocity is common at all receivers and this finally lead to same estimated layer velocity and depth by recalling (4.13) and (4.14).



(a) Velocity estimation



(b) Depth estimation

Figure 4.5: Centralized NMO estimation performance

4.1.3 The effect of inter-receiver distance

An interesting question in classic Normal-moveout without recalculating picking time is how the inter-receiver distance influences estimated results? Therefore, we come to investigate the effect of inter-receiver distance. Firstly, we place 6 receivers on one side with different inter-receiver distance. Then we define an average velocity and depth estimation error as:

$$\hat{\epsilon}_v = \frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\hat{v}_{d,f}-v_{d,f}|}{|v_{d,f}|}}{Df_r} \tag{4.28}$$

$$\hat{\epsilon}_d = \frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\hat{h}_{d,f}-h_{d,f}|}{|h_{d,f}|}}{Df_r} \tag{4.29}$$

where f_r is number of layers. $\hat{v}_{d,f}$ and $\hat{h}_{d,f}$ are estimated layer velocity and depth for dth receiver at fth layer. $v_{d,f}$ and $h_{d,f}$ are true layer velocity and depth for dth receiver at fth layer. Then we could choose different inter-receiver distance to replace receivers and observe how it influences estimation performance, which is shown upper subplot in Fig 4.6. it is obvious that estimation error is show an overall increasing trend with increasing inter-receiver distance but is worst at inter-receiver distance of 1.2m. This could be explained as follows: instead of directly analyze the estimation error of internal layer velocity $\hat{\epsilon}_v$, we could come to analyze how the inter-receiver distance influences the average normalized root-mean square velocity estimation error e_{rms-v} in NMO equation. This is because if $\hat{\epsilon}_{rms-v}$ increases which would also means $\hat{\epsilon}_v$ increases and internal layer velocity is calculated recursively from root-mean square velocity and therefore it is hard

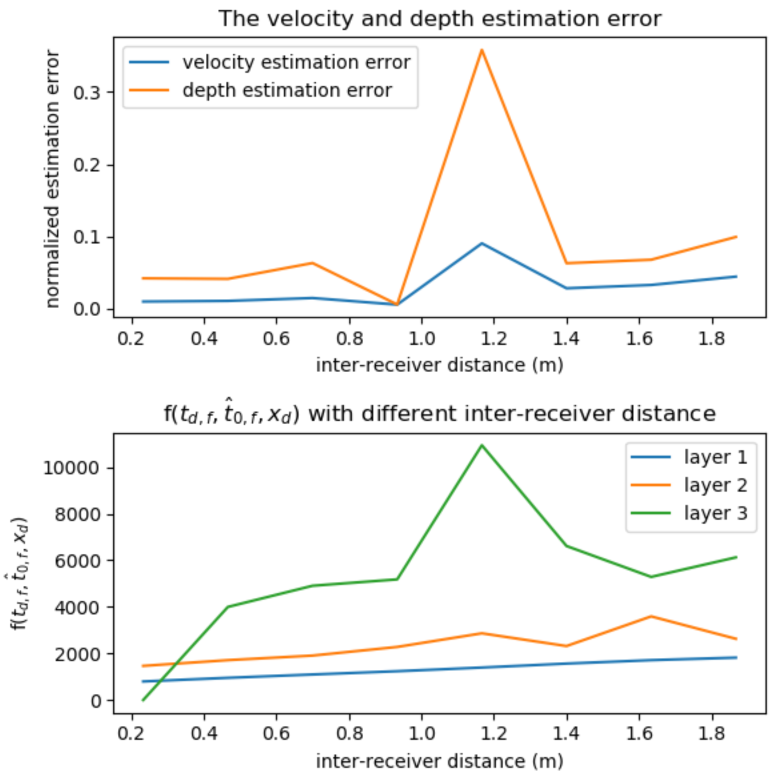


Figure 4.6: Receiver distance effect analysis

to analyze it mathematically. Thus, we define ϵ_{rms-v} as:

$$\epsilon_{rms-v} = \frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\hat{v}_{d,f}^{rms} - v_{d,f}^{rms}|}{|v_{d,f}^{rms}|}}{Df_r} \tag{4.30}$$

From normal moveout equation we know for estimated root mean square velocity at fth layer is given as:

$$\hat{v}_{d,f}^{rms} = \frac{x_d}{\sqrt{(t_{d,f}^2 - \hat{t}_{0,f}^2)}} \tag{4.31}$$

and the ground truth of root mean square velocity is defined as $v_{d,f}^{rms}$ Therefore, ϵ_{rms-v} can be rewritten as:

$$\begin{aligned}\epsilon_{rms-v} &= \frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\frac{x_d}{\sqrt{(t_{d,f}^2 - \hat{t}_{0,f}^2)}} - v_{d,f}^{rms}|}{|v_{d,f}^{rms}|}}{Df_r} \\ &= \frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}|\frac{x_d}{\sqrt{(t_{d,f}^2 - \hat{t}_{0,f}^2)}v_{d,f}^{rms}} - 1|}{Df_r}\end{aligned} \tag{4.32}$$

Since ground truth remains constant for same layer and therefore we could come to analyze the term for data at given layer:

$$f(t_{d,f}, t_{0,f}, x_d) = \sum_{d=1}^{D}|\frac{x_d}{\sqrt{(t_{d,f}^2 - \hat{t}_{0,f}^2)}}| \tag{4.33}$$

The pattern of $f(t_{d,f}, t_{0,f}, x_d)$ would generally reflects variation of average normalized root mean square velocity error. This is shown in subplot at the bottom of Fig 4.6, where we observe it show an overall increasing trend for all three layers and has prominent peak at inter-receiver distance of 1.2m for third layer shown as green curve. This implies picking is relatively much worse with this inter-receiver distance in an average sense. Additionally, we could see normalized averaged estimation error of t_0 is defined in 4.34.

$$g(t_{0,f}) = \frac{1}{f_r D}\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\hat{t}_{0,f} - t_{0,f}|}{t_{0,f}} \tag{4.34}$$

where $t_{0,f}$ is ground truth t_0 for fth layer.

It is increasing with inter-receiver distance in Fig 4.8, as larger distance between receivers means data points are further away from each other and make nonlinear least-square fitting harder to get accurate estimated results. Combine with picking error, finally produce velocity estimation error curve variation in Fig 4.6 in this case. For averaged depth estimation error $\hat{\epsilon}_r$, recall (4.14) , therefore it shows similar pattern as $\hat{\epsilon}_v$ but smaller as estimated thickness at fth layer for each receiver by $\frac{1}{2}(\hat{t}_{0,f} - \hat{t}_{0,f-1})$. One **important point** is the larger distance between receivers doesn't necessarily mean a worse estimation result at receiver in classic centralized NMO. As we observe estimation performance is oscillating at different receivers from Fig 4.4 that shows some receivers have better performance with larger inter-receiver distance, which seems contract our conclusion. This is because t_0 estimate and receiver offset and picking travel time error will change when we modify the inter-receiver distance, but we use same t_0 estimate in centralized NMO algorithm for all receivers.

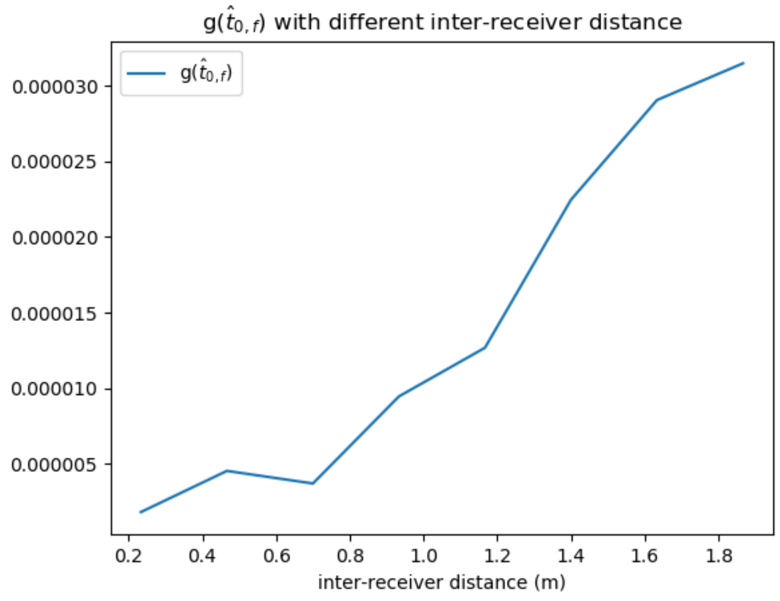


Figure 4.7: Normalized averaged estimation error $g(t_{0,f})$

4.2 Distributed NMO algorithms

The aforementioned centralized NMO algorithm relies on information access on all receivers, which is problematic in realistic scenarios, as communication delay and interference through physical channel between receivers might be blocked for distant receivers. Additionally, it requires much time to finish estimation in the whole receiver network, as each receiver need to firstly gather information to perform NMO algorithm and then loop over others receivers. Therefore, the state-of-the-art distributed NMO algorithms should be taken into consideration to solve estimation task via NMO based on data exchange between receivers and their neighbors in a completely distributed manner. Before design of corresponding distributed algrothms, some basic assumptions should be made in the distributed setting.

- Each receiver has perfect knowledge about its distance with respect to source.
- Each receiver only has access to its neighbors and can transmit data in between.

4.2.1 Distributed NMO with average consensus

As in the Normal-moveout scheme, the parameters slowness u_f and two-way vertical travel time $t_{0,f}$ are same for one layer from analytical model of NMO in (4.16) and we would obtain same estimated parameters from centralized NMO algorithm at all receivers. In distributed scenario, it is also feasible to estimated parameters via nonlinear least-square at different receivers with picking travel time sharing between receivers and their neighbors. However, estimated parameters would differ from each other, as each receiver gathers different picking time for fitting. Therefore, it is crucial to design algorithm that enforce all receivers consensus on a set of specific parameters. One point to be noted is we don't consider noisy link between receivers for following algorithm development. With this motivation, the distributed average consensus algorithm is introduced to achieve asymptotic average consensus with any initial estimated parameters at each receiver. To formulate this algorithm, we firstly consider a network with D receivers. We also denote $\hat{t}^k_{0,d,f}$ as estimated t_0 for fth layer of dth receiver at time step k. The edge of receiver d and i is denoted as (d,i). The dth receiver has an edge set denoted as $\mathcal{N}_i$, which contains all its neighbors. The edge (d,i) is associated with a weight denote as $W_{d,i}$. Here we implement this algorithm with respect to all individual estimated parameters $\hat{t}_{0,d,f}$ and $\hat{u}_{0,d,f}$, which are two parameters govern analytic model of normal moveout. If dth receiver has L_d neighbours. The element in weight matrix is given as follows:

$$W_{d,i} = \begin{cases} \frac{1}{L_d+1} & i \in \mathcal{N}_d \\ 0 & i \notin \mathcal{N}_d \end{cases} \tag{4.35}$$

$$W_{d,d} = 1 - \frac{L_d}{L_d+1} \tag{4.36}$$

This weight coefficient assignment is easy to interpret as each receiver delivers $\frac{1}{L_d+1}$ portion of its data to one of its neighbours. After delivering for all neighbours, there would be $1 - \frac{L_d}{L_d+1}$ portion of data remaining in itself.

The update recursion of receiver:

$$\hat{t}^{k+1}_{0,d,f} = \hat{t}^k_{0,d,f} + \sum_{i \in \mathcal{N}_d} W_{d,i}(\hat{t}^k_{0,d,f} - \hat{t}^k_{0,i,f}) \tag{4.37}$$

$$\hat{u}_{0,d,f}^{k+1} = \hat{u}_{0,d,f}^{k} + \sum_{i \in \mathcal{N}_d} W_{d,i}(\hat{u}_{0,d,i}^{k} - \hat{u}_{0,d,i}^{k}) \tag{4.38}$$

Both (4.37) and (4.38) represents a linear iterative update process and can be interpreted as follows: at each step, each receiver collect estimated parameters from its neighbors and then calculate the difference of estimated parameters between their neighbors and themselves. Afterwards, this difference is multiplied with associated edge weight with respect to each neighbor and each receiver sum over these weighted difference and plus estimated parameters at current time step. This final result is the updated estimated parameters at next step. (4.37) and (4.38) can also be written in terms of weighting matrix $\mathbf{W}$ as:

$$\hat{\mathbf{t}}_{0,f}^{k+1} = \mathbf{W}\hat{\mathbf{t}}_{0,f}^{k} \tag{4.39}$$

$$\hat{\mathbf{u}}_{f}^{k+1} = \mathbf{W}\hat{\mathbf{u}}_{f}^{k} \tag{4.40}$$

A nature question is what the condition of weight matrix to get asymptotic average consensus? If we denote a vector with all ones as $\mathbf{l}$. The asymptotic average consensus implies:

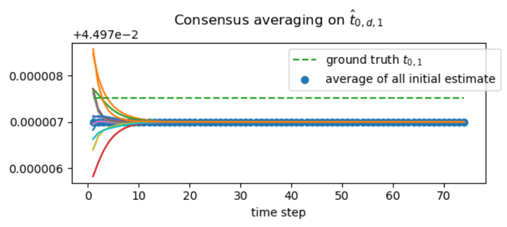
$$\lim_{k \to \infty} \mathbf{W}^k \hat{\mathbf{t}}_f^0 = \frac{1}{D}\mathbf{l}\mathbf{l}^T \hat{\mathbf{t}}_f^0 \tag{4.41}$$

Where $\hat{\mathbf{t}}_f^0 = [\hat{\mathbf{t}}_{0,f}, \hat{\mathbf{t}}_{0,f}, \ldots, \hat{\mathbf{t}}_{D,f}]^T$ contains estimated t_0 at all receivers for fth layer.

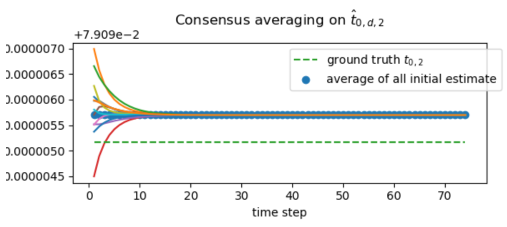
The condition governs this limitation is given in (4.42) 9:

$$||\mathbf{W} - \mathbf{Q}||_2 < 1 \tag{4.42}$$

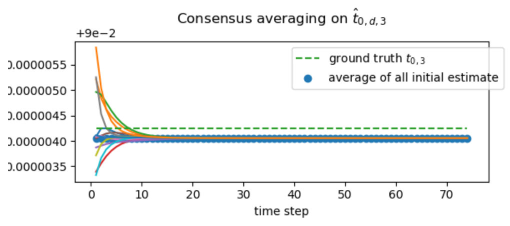
Here $||\ldots||_2$ is denoted as spectral norm and $\mathbf{Q} = \frac{1}{D}\mathbf{l}\mathbf{l}^T$. To observe parameters update with average consensus, we apply simulation for the same receiver network in previous centralized algorithm but using a random receiver topology with $L_d = 4$ neighbors for each receiver, which means each receiver randomly choose 4 neighbors in whole receiver network to exchange data. The simulation results are shown in Fig 4.8 (a)-(f). The dashed green line in these 6 subplots indicate corresponding ground truth parameters. Different solid lines in the subplot indicate estimated parameter curve for each receiver. Obviously, initially different receiver has different estimated parameter. However, estimated parameters among receivers are being more and more similar via parameter sharing with their neighbors and finally all receivers consensus on average of all initial estimate shown as dots in the figure. Meanwhile, estimated slowness $\hat{u}_{d,f}$ is relatively further and further away from ground truth with deeper layer shown in Fig 4.8 (d)-(f). This is because slowness is extremely small compare with two-way vertical travel time estimate $\hat{t}_{0,d,f}$ and it is smaller with deeper layer. Then, nonlinear least-square fitting would show worse performance to fit for slowness with deeper layer, as its smaller value makes it plays less and less important role in minimizing sum of squares error in the fitting procedure. On the contrary, $\hat{t}_{0,d,f}$ is more and more important on this minimization with deeper layer and thus we can see $\hat{t}_{0,d,f}$ for 3rd layer is relatively much closer than two upper layers by comparing Fig 4.8 (a)-(c).



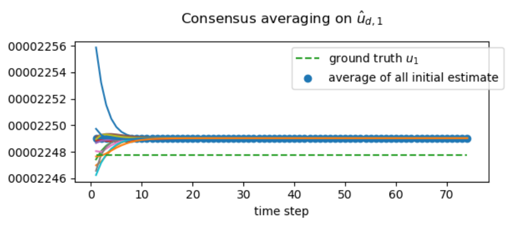
(a) estimated $\hat{t}_{0,d,1}$



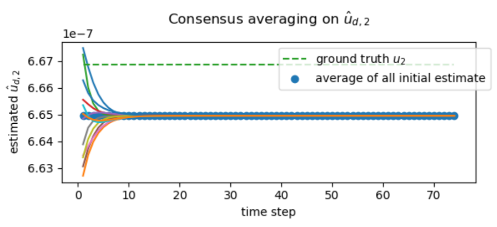
(b) estimated $\hat{t}_{0,d,2}$



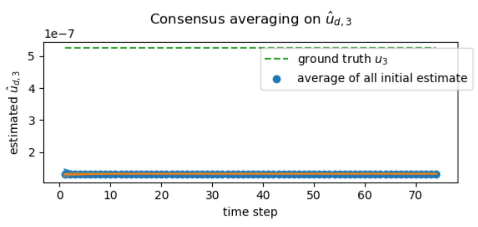
(c) estimated $\hat{t}_{0,d,3}$



(d) estimated $\hat{u}_{d,1}$



(e) estimated $\hat{u}_{d,2}$



(f) estimated $\hat{u}_{d,3}$

Figure 4.8: Average consensus on $\hat{t}_{0,d,f}$ and $\hat{u}_{d,f}$

Distributed NMO with average consensus algorithm structure

On the basis of average consensus, a state-of-the-art distributed NMO algorithm is formulated in Algorithm 5. Before explain the algorithm, we firstly define some notations used in its pseudocode: $\mathbf{t}_{d,n}^{N}$ is denoted as picking travel time of nth neighbor of dth receiver for all layers and $x_{d,n}^{N}$ is source-to-receiver distance nth neighbor of dth receiver. The whole algorithm flow is explained as follows: firstly each receiver will take a noisy seismic measurment $\mathbf{z}_{d}^{n}$ and then we perform picking on travel time either based on deconvolved measurement or from STA/LTA picking algorithm to get picking at all receivers for each layer, correspond to line 1-8. Line 9-13 represents a distributed nonlinear least-square fitting process that each receiver gathers picking travel time and source-to-receiver position from its neighbors and perform fitting to get initial estimate parameters $\hat{t}_{0,d,f}^{1}$ and $\hat{u}_{d,f}^{1}$. Afterwards, we enter into inner loop of average consensus update in line 16-17. At step k, we perform a recalculation of picking travel time at receivers based on their current estimate parameters. Simultaneously, each receiver would also perform a local estimation of layer velocity and thickness via Normal-moveout, correspond to line 18-25. At the end of algorithm, each receiver would consensus on a set of parameters $[\hat{t}_{0,d,f}^{k_{max}}, \hat{u}_{d,f}^{k_{max}}]^{T}$ and obtain corresponding converged estimated layer velocity and thickness $\hat{h}_{d,f}^{k_{max}}$ and $\hat{v}_{d,f}^{k_{max}}$. The superiority of this distributed NMO algorithm compared with previous centralized one is each receiver has the ability to perform estimation with its local available information in parallel, while only central node could perform estimation in centralized NMO. This makes whole receiver network much more flexible and robust, as main functionality of receiver network is maintained even though some receivers failed at given moment.

Estimation performance analysis

From consideration of distance factors in communication among receivers, the line topology of neighbor arrangement in distributed receiver network should be considered which can be explained with Fig 4.9. In this figure, there are given number of receivers equally and symmetrically spaced on both sides of source and receiver located with coordinate x_d has a communication radius R_c that allows it can only communicate with its nearest receivers defined by number of neighbors shown as red dots, where others are blocked shown as black dots.

As receivers are equally spaced on both sides of source which turns out if dth receiver has L_d neighbors in theory where L_d should be an even number with this communication radius definition, then we could formulate neighbor assignment for dth receiver in (4.43):

$$L_d^a = \begin{cases} L_d - 1 & d - \frac{L_d}{2} <= \frac{D}{2} + 1 <= d + \frac{L_d}{2} \\ D - d + \frac{L_d}{2} & D - \frac{L_d}{2} < d <= D \\ d + \frac{L_d}{2} & d < \frac{L_d}{2} \\ L_d & else \end{cases} \tag{4.43}$$

Algorithm 5 Distributed NMO with average consensus

```
1:  for $\mathbf{z}_d^n$ d=1,...,D: do
2:      choose picking method
3:      if picking from deconvolved measurement then
4:          $\hat{\boldsymbol{\mu}}_d = (\frac{\mathbf{W}^T\mathbf{W}}{\sigma^2} + \lambda\mathbf{I})^{-1}\frac{\mathbf{W}^T\mathbf{z}_d^n}{\sigma^2}, \lambda = \frac{1}{\sigma^3}$
5:          travel-time picking
6:      else
7:          picking from STA/LTA picker
        end if
8:      $\mathbf{t} = [\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_{f_r}]^T$
    end for
9:  for receiver $d = 1, \ldots, D$: do
10:     gather arrival time estimate and receiver offset from its $L_d$ neighbours:
11:     $\mathbf{t}_d^N = [\mathbf{t}_{d,1}^N, \mathbf{t}_{d,2}^N, \ldots, \mathbf{t}_{d,L_d}^N]^T$
12:     $\mathbf{x}_d^N = [x_{d,1}^N, x_{d,2}^N, \ldots, x_{d,L_d}^N]^T$
13:     Nonlinear least-square fitting: $\rightarrow \hat{t}_{0,d,f}^1, \hat{u}_{d,f}^1, f = 1, \ldots, f_r$
    end for
14: for iteration $k = 1, 2, \ldots, k_{max}$: do
15:     for layer $f = 1, \ldots, f_r$: do
16:         $\hat{\mathbf{t}}_{0,f}^{k+1} = \mathbf{W}\hat{\mathbf{t}}_{0,f}^k$
17:         $\hat{\mathbf{u}}_f^{k+1} = \mathbf{W}\hat{\mathbf{u}}_f^k$
18:         recalculate time estimate:
19:         for receiver $d = 1, \ldots, D$: do
20:             $\hat{t}_{d,f}^2 = \hat{u}_{d,f}^k x_d^2 + {\hat{t}_{0,d,f}^k}{}^2$
21:             Perform NMO locally
22:             reconstruct velocity and depth:
23:             $\hat{v}_{d,f}^{rms-k} = \frac{x_d}{\sqrt{(\hat{t}_{d,f}^2 - {\hat{t}_{0,d,f}^k}{}^2)}}$
24:             $\rightarrow \hat{v}_{d,f}^k = \sqrt{\frac{(\hat{v}_{d,f}^{rms-k})^2\hat{t}_{0,d,f}^k - (\hat{v}_{d,f-1}^{rms-k})^2\hat{t}_{0,d,f-1}^k}{\hat{t}_{0,d,f}^k - \hat{t}_{0,d,f-1}^k}}$
25:             $\rightarrow \hat{h}_{d,f}^k = \frac{1}{2}(\hat{t}_{0,d,f}^k - \hat{t}_{0,d,f-1}^k)\hat{v}_{d,f}^k$
            end for
        end for
    end for$\rightarrow \hat{t}_{0,d,f}^{k_{max}}, \hat{u}_{d,f}^{k_{max}}, \hat{h}_{d,f}^{k_{max}}, \hat{v}_{d,f}^{k_{max}}$
```

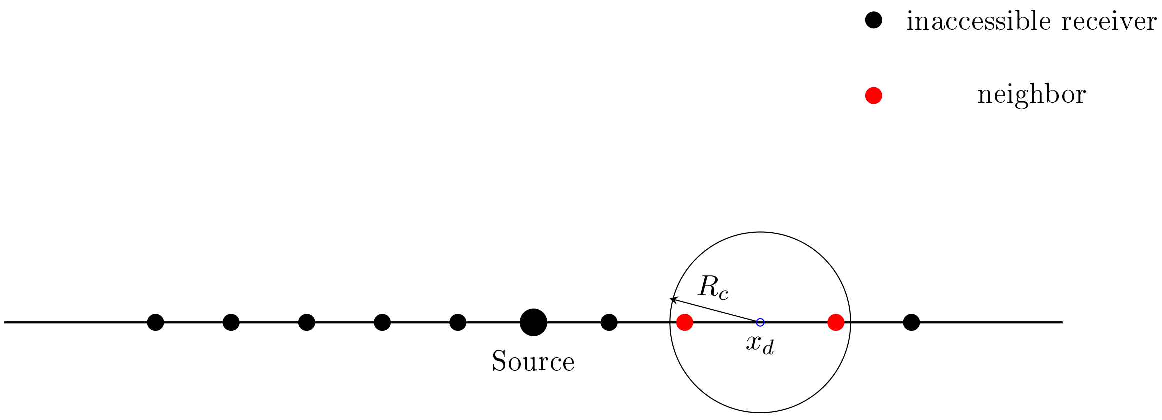


Figure 4.9: Line topology illustration

Where L_d^a is actual number of neighbors for dth receiver. From neighbor assignment we could discover the receiver closer to left or right boundaries will have less neighbors when its communication circle region cover the boundary and the leftmost and rightmost have least number of neighhbors. Also, when communication region of one receiver covers source, then it would only have $L_d - 1$ neighbors. Then we apply line topology in distributed NMO with average consensus algorithm with given simulation setup in Table 4.1. The velocity and depth estimation results at different iterations are shown in Fig 4.10 and Fig 4.11. From observation of four subplots in Fig 4.10, it is clearly that initially each receiver has different estimated result on one side, because they have different estimated parameters from fitting with gathering of neighbouring picking travel time. Then the whole estimation performance is becoming better and better and at iteration 37 we can observe all receivers have already converged, which show a very good ultimate estimation result. Interestingly, not all receivers improve their performance during averaging consensus with repicking which can be seen by comparing subplot (a) and (b) in Fig 4.10. For receivers at around 5m away from source is very good initially but being much worse at 6th iteration. This is because each update in average consensus associate with information incorporation of neighbors. Then, with line topology these receiver would firstly incoporate estimated parameters with nearest receiver which have very bad estimation performance implies bad estimate parameters shown in subplot (a). As a consequence, at first several iterations they would average over these bad estimated parameters and get a worse estimation performance than before. However, each receiver would average over estimated parameters from more and more receivers with algorithm update and therefore influence from initial bad estimated parameters is being smaller and smaller and all receivers would have more and more similar estimated results and finally converge, when all receivers consensus on averge of all initial estimated parameters. This corresponds to procedure in subplots (b)-(d). The depth estimation performance in Fig 4.11 follows same pattern as in the velocity estimation but with smaller variation, as depth estimated result at dth receiver for fth layer calculated from ground is obtained by multiplying with local estimated two-way vertical travel time $\hat{t}_{0,d,f}$ and $\hat{t}_{0,d,f}$ is smaller than 1 in our scenario.

Table 4.1: Simulation setup for distributed NMO with average consensus

length X	**number of receivers**	**number of neighbors L_d**	**SNR**
28 m	24	4	5dB

Comparison with centralized NMO

After the incorporation of distributed NMO with average consensus, we could compare its performance with previous centralized case. First of all, the normalized average velocity and depth estimation error should be defined for distributed and centralized case as E_v,E_d,E_v^c and E_d^c which are in logarithmic scale shown in (4.44)-4.47:

$$E_v = \log\left(\frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\hat{v}_{d,f}-v_{d,f}|}{|v_{d,f}|}}{Df_r}\right) \tag{4.44}$$

$$E_d = \log\left(\frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\hat{h}_{d,f}-h_{d,f}|}{|h_{d,f}|}}{Df_r}\right) \tag{4.45}$$

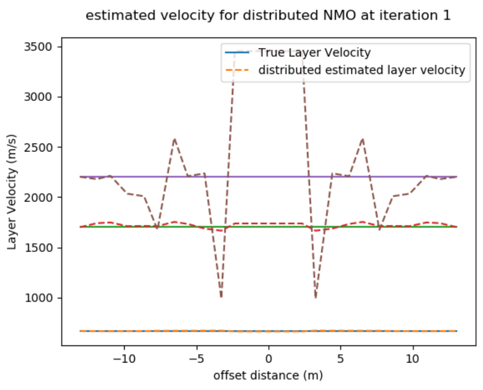
$$E_v^c = \log\left(\frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\hat{v}_{d,f}^c-v_{d,f}|}{|v_{d,f}|}}{Df_r}\right) \tag{4.46}$$

$$E_d^c = \log\left(\frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\hat{h}_{d,f}^c-h_{d,f}|}{|h_{d,f}|}}{Df_r}\right) \tag{4.47}$$

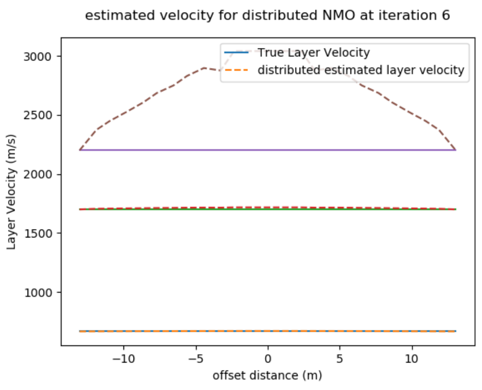
Their estimation comparison is plotted in Fig 4.12, where we observe there exists a very small bias between distributed and centralized schemes after convergence. This can be stated mathematically as follows. From average consensus perspective, each receiver would agree on average of all initial estimate parameters:

$$\hat{t}_{0,d,f}^{k_{max}} = \frac{1}{D}\sum_{d=1}^{D}\hat{t}_{0,d,f}^{1} \tag{4.48}$$

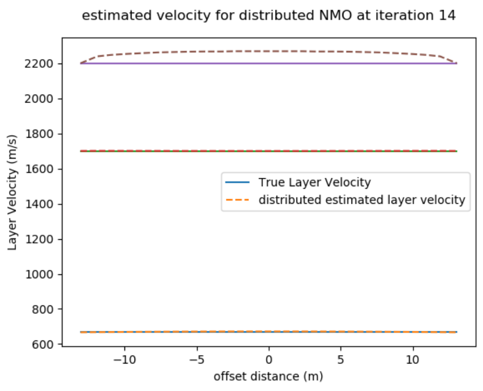
$$\hat{u}_{d,f}^{k_{max}} = \frac{1}{D}\sum_{d=1}^{D}\hat{u}_{d,f}^{1} \tag{4.49}$$



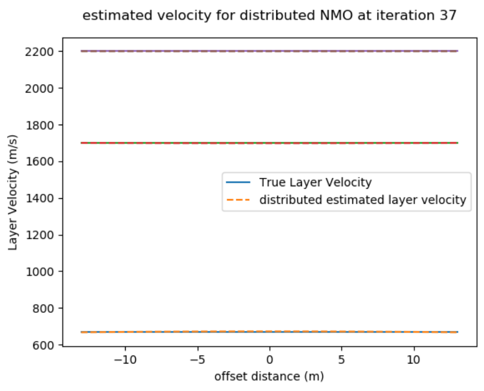
(a) initial estimation



(b) 6th iteration

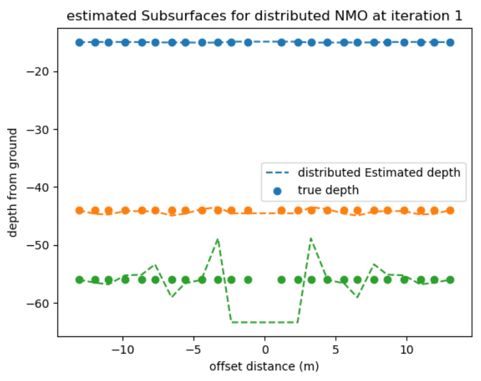


(c) 14th iteration

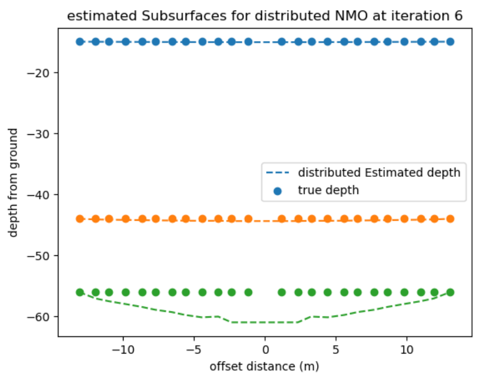


(d) 37th iteration

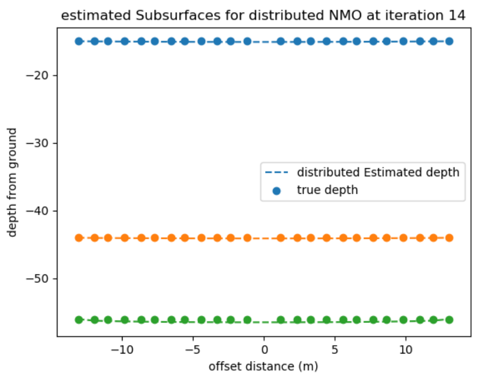
Figure 4.10: Velocity estimation results for distributed NMO with average consensus



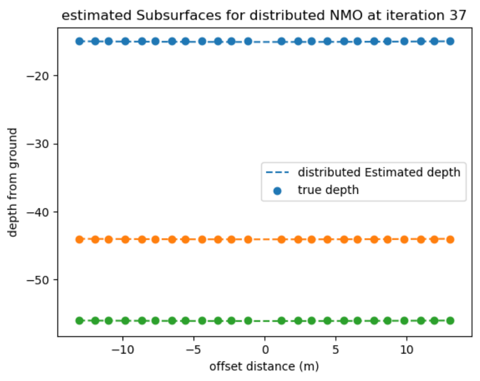
(a) initial estimation



(b) 6th iteration



(c) 14th iteration



(d) 37th iteration

Figure 4.11: Depth estimation results for distributed NMO with average consensus

For centralized case, we perform fitting to obtain estimated parameters $\hat{t}^c_{0,d,f}$ and $\hat{u}^c_{d,f}$ and therefore the bias between two schemes will be proportional to the difference of their final estimated parameters after convergence.

$$|\hat{t}^{k_{max}}_{0,d,f} - \hat{t}^c_{0,d,f}| = |\frac{1}{D}\sum_{d=1}^{D} \hat{t}^1_{0,d,f} - \hat{t}^c_{0,d,f}| \tag{4.50}$$

$$|\hat{u}^{k_{max}}_{d,f} - \hat{u}^c_{d,f}| = |\frac{1}{D}\sum_{d=1}^{D} \hat{u}^1_{d,f} - \hat{u}^c_{d,f}| \tag{4.51}$$

As we use picking from deconvolved profile to perform nonlinear least-square fitting which fits the pattern of ground truth very well shown in Fig 3.8 (b). Thus, estimated results with picking time from neighbors in distributed case is very close to that from picking time with all receivers in centralized case. This turns out final averaged consensused parameters are also very close to centralized one which results a very small bias between two schemes in Fig 4.12.

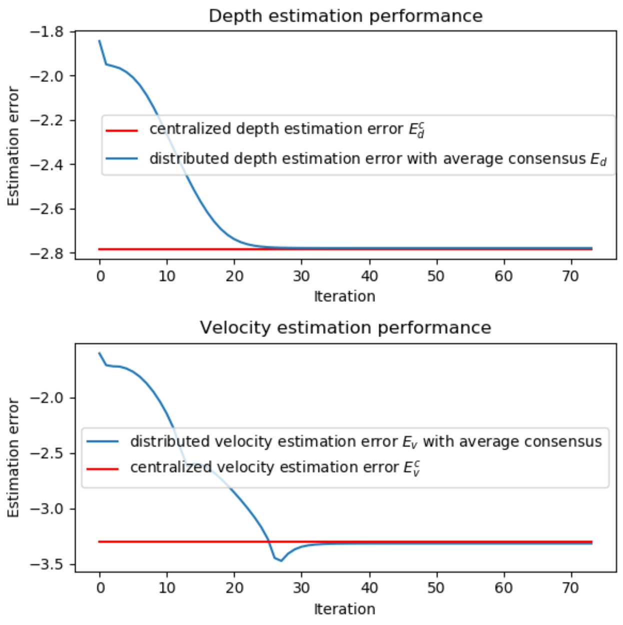
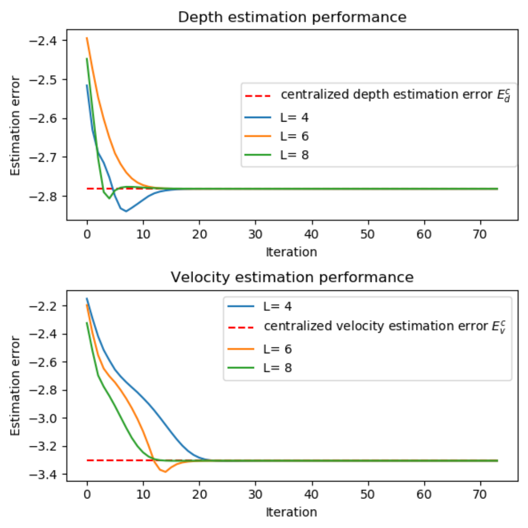


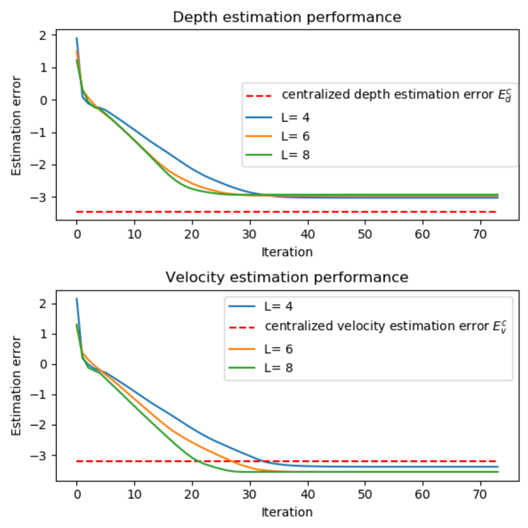
Figure 4.12: The estimation performance comparison between centralized NMO and distributed NMO with average consensus

The effect of neighbor number

An interesting question involving the distributed NMO algorithm with average consensus is how the number of neighbors influences estimation performance. Therefore, we use different number of neighbors to simulate how estimation curve varies shown in Fig 4.13 (a)-(b). The left subplot of Fig 4.13 we use picking obtained from picking algorithm processed on deconvolved measurement and we simulate estimation performance for neighbor number $L = 4, 6, 8$. On the right hand side, same simulations are performed except that we use picking from STA/LTA picker. Clearly, the slope of estimation curve is steeper and converge faster with increasing number of neighbors. This is because initial estimated parameters will be better with larger number of neighbors from data fitting perspective, as each receiver gather more data points for fitting. This is also shown in the figure. In average consensus, each receiver exchange data with their neigbors and thus larger number of neigbors will accelerate information diffusion in the network and enforce faster consensus on parameters. Additionally, theres is much large bias between distributed and centralized NMO performance with picking from STA/LTA picker. Because picking is fluctuating relatively rapid and doesn't show a hyperbolic pattern with STA/LTA picker, which cause local estimated parameters at different receiver differ much away from each other. Then recall (4.50)-(4.51), final average consensused parameters also would deviate much away from estimated parameters in centralized case and results in large estimation error bias in steady-state.



(a) picking from deconvolved profile



(b) picking from STA/LTA picker

Figure 4.13: The influence of neighbor number on estimation

4.2.2 Distributed NMO with Adapt-then-combine

In addition to previous distributed NMO scheme with average consensus, we may ask if we could solve this distributed estimation regarding to layer velocity and depth in alternate way. This motivates us to link estimation task with travel-time tomography. The travel-time tomography aims to measure travel time of wave to inversely reconstruct velocity field distribution of subsurface model with these measurement. To generate synthetic measured travel time, a typical method is to solve Eikonal equation. However, with our simple horizontal subsurface models, we could use picking travel time from previous chapter as measured travel time of seismic reflected ray. By combining them with Normal-moveout model in (4.15), we can apply the adapt-then-combine diffusion learning to estimate layer velocity and depth in distributed fashion and also we don't consider noisy link between receivers for following algorithm development.

Motivation and Stochastic gradient descent

Firstly, we come to model of measured travel time and synthesized travel time of reflection wave, Typically, for fth layer and a receiver network consisting of D receivers, synthesized travel time can be described by a NMO model with estimated parameters in distributed NMO with adapt-then-combine are $\tilde{u}_f$ and $\tilde{t}_f$:

$$\tilde{\mathbf{t}}_f = \sqrt{\tilde{u}_f \mathbf{x}^2 + \tilde{t}_{0,f}^2} \tag{4.52}$$

Where $\tilde{\mathbf{t}}_f = [\tilde{t}_{1,f}, \ldots, \tilde{t}_{D,f}]^T$ and $\mathbf{x} = [x_1, x_2, \ldots, x_D]^T$ represent travel time for different receivers and receiver position respectively. Then this model enables us to apply gradient-based methods like stochastic gradient descent (SGD), quasi-newton method, conjugate gradient descent to find optimal synthesized parameters fit our synthesized travel-time data. After that, we can directly calculate depth and layer velocity based on estimated parameters from SGD. For standard SGD implementation, we need to define a cost function which describes the difference between synthesized travel time $\tilde{t}_{d,f}$ and measured travel time $\mathbf{t}_{d,f}$. The global cost function in the receiver network is given in (4.53), which is summation of local cost at each receiver.

$$C_g(\tilde{u}_{d,f}, \tilde{t}_{0,d,f}) = \sum_{d=1}^{D} c_d(\tilde{u}_{d,f}, \tilde{t}_{0,d,f}) = \sum_{d=1}^{D} (\tilde{t}_{d,f} - t_{d,f})^2 \tag{4.53}$$

However, our network is distributed arranged and thus each receiver only has limited amount of information from its neighbours. Therefore, in the distributed scenario, each receiver would take its own measured travel time. Based on these data, they perform stochastic gradient descent and we define an individual cost function for each receiver as:

$$c_d(\tilde{u}_{d,f}, \tilde{t}_{0,d,f}) = (\tilde{t}_{d,f} - t_{d,f})^2 \tag{4.54}$$

The gradient of cost function of each receiver with respect to $\tilde{u}^1_{d,f}$ is calculated in (4.55):

$$\begin{aligned}\frac{\partial c_d(\tilde{u}_{d,f},\tilde{t}_{0,d,f})}{\partial \tilde{u}_{d,f}} &= \frac{\partial(t_{d,f}-\tilde{t}_{d,f})^2)}{\partial \tilde{u}_{d,f}} \\ &= 2(t_{d,f}-\tilde{t}_{d,f})\frac{\partial(t_{d,f}-\tilde{t}_{d,f})}{\partial \tilde{u}_{d,f}} \\ &= 2((t_{d,f}-\sqrt{\tilde{u}_{d,f}{x_d}^2+\tilde{t}^2_{0,d,f}}))\frac{\partial((t_{d,f}-\sqrt{\tilde{u}_{d,f}{x_d}^2+\tilde{t}^2_{0,d,f}}))}{\partial \tilde{u}_{d,f}} \\ &= ((t_{d,f}-\sqrt{\tilde{u}_{d,f}{x_d}^2+\tilde{t}^2_{0,d,f}}))\frac{-x_d^2}{\sqrt{\tilde{u}_{d,f}{x_d}^2+\tilde{t}^2_{0,d,f}}} \\ &= (1-\frac{t_{d,f}}{\sqrt{\tilde{u}_{d,f}{x_d}^2+\tilde{t}^2_{0,d,f}}})x_d^2\end{aligned} \tag{4.55}$$

Similarly, gradient of cost function with respect to parameter $\tilde{t}_{0,d,f}$ is given in (4.56):

$$\begin{aligned}\frac{\partial c_d(\tilde{u}_{d,f},\tilde{t}_{0,d,f})}{\partial \tilde{t}_{0,d,f}} &= \frac{\partial((t_{d,f}-\tilde{t}_{d,f})^2)}{\partial \tilde{t}_{0,d,f}} \\ &= 2(\tilde{t}_{d,f}-t_{d,f})\frac{\partial(\tilde{t}_{d,f}-t_{d,f})}{\partial \tilde{t}_{0,d,f}} \\ &= 2((t_{d,f}-\sqrt{\tilde{u}_{d,f}{x_d}^2+t^2_{0,d,f}}))\frac{\partial((\tilde{t}_{d,f}-\sqrt{\tilde{u}_{d,f}{x_d}^2+t^2_{0,d,f}}))}{\partial \tilde{t}_{0,d,f}} \\ &= ((t_{d,f}-\sqrt{\tilde{u}_{d,f}{x_d}^2+\tilde{t}^2_{0,d,f}}))\frac{-2\tilde{t}_{0,d,f}}{\sqrt{\tilde{u}_{d,f}x_d^2+\tilde{t}^2_{0,d,f}}} \\ &= 2\tilde{t}_{0,d,f}(1-\frac{t_{d,f}}{\sqrt{\hat{u}_{d,f}{x_d}^2+\tilde{t}^2_{0,d,f}}})\end{aligned} \tag{4.56}$$

The SGD is implemented as follows: given observed synthesized travel time, we start with an initial guess of parameter $\tilde{u}^1_{d,f}$ and $\tilde{t}^1_{0,d,f}$. Then for each iteration we could perform SGD update according to above two gradients and a learning rate β in 4.57 and 4.58:

$$\tilde{u}^{k+1}_{d,f} = \tilde{u}^k_{d,f} - \beta\frac{\partial c_d(\tilde{u}^k_{d,f},\tilde{t}^k_{0,d,f})}{\partial \tilde{u}_{d,f}} \tag{4.57}$$

$$\tilde{t}^{k+1}_{0,d,n} = \tilde{t}^k_{0,d,f} - \beta\frac{\partial c_d(\tilde{u}^k_{d,f},\tilde{t}^k_{0,d,f})}{\partial \tilde{t}_{0,d,f}} \tag{4.58}$$

The learning rate should be chosen appropriately to ensure convergence to reach global optimal paramters if cost function is strictly convex . As we can see from 4.57 and 4.58, updated parameters moves in the opposite direction of cost function gradient, thus it will approach to global minimum point for a strictly convex cost function . If cost function is non-convex, SGD update may stuck at local minimum or saddle point. Also, a large learning rate may cause updated weight move across global minimum and then it would deviates from minimum more and more, which causes calculated cost function increases infinitely.

Choice of learning rate

Firstly, we summarize the parameters into a parameter vector as: $\tilde{\mathbf{p}}_d^k = [\tilde{u}_{d,f}^k, \tilde{t}_{0,d,f}^k]^T$. Therefore, the SGD update could be written as:

$$\tilde{\mathbf{p}}_d^{k+1} = \tilde{\mathbf{p}}_d^k - \beta \frac{\partial c_d(\mathbf{b}_d^k)}{\partial \mathbf{b}_d^k} \tag{4.59}$$

The cost function can be approximated at optimum point with Taylor expansion [4]:

$$c_d(\tilde{\mathbf{p}}_d^k) \approx c_d(\tilde{\mathbf{p}}_d^o) + \frac{1}{2}(\tilde{\mathbf{p}}_d^k - \mathbf{p}_d^o)^T \mathbf{H}_{\mathbf{p}_d^o}(\tilde{\mathbf{p}}_d^k - \mathbf{p}_d^o) \tag{4.60}$$

where $\mathbf{p}_d^o$ is optimum parameter which has a zero gradient of cost function at this point. Then, we can replace derivative with Hessian matrix $\mathbf{H}_{\mathbf{p}_d^o}$ associated with parameters evaulated at optimum parameters $\mathbf{p}_d^o$:

$$\tilde{\mathbf{p}}_d^{k+1} \approx \tilde{\mathbf{p}}_d^k - \beta \mathbf{H}_{\mathbf{p}_d^k}(\tilde{\mathbf{p}}_d^k - \mathbf{p}_d^o) \tag{4.61}$$

Then we subtract both sides of (4.61) with $\mathbf{p}_d^o$ as:

$$\tilde{\mathbf{p}}_d^{k+1} - \mathbf{p}_d^o \approx (\mathbf{I} - \beta \mathbf{H}_{\mathbf{p}_d^o})(\tilde{\mathbf{p}}_d^k - \mathbf{p}_d^o) \tag{4.62}$$

We can repeat this recursion for many times and get:

$$\tilde{\mathbf{p}}_d^{k+1} - \mathbf{p}_d^o = (\mathbf{I} - \beta \mathbf{H}_{\mathbf{p}_d^o})^k(\tilde{\mathbf{p}}_d^1 - \mathbf{p}_d^o) \tag{4.63}$$

Then we use spectral decomposition to substitute Hessian matrix as:

$$\mathbf{H}_{\mathbf{p}_d^o} = \mathbf{\Gamma}\mathbf{\Sigma}\mathbf{\Gamma}^T \tag{4.64}$$

Where $\mathbf{\Gamma}$ is an orthogonal matrix and $\mathbf{\Sigma}$ is diagonal matrix contains eigenvalues ε_i of Hessian matrix for ith row. Then take it into (4.63) as:

$$\begin{aligned} \tilde{\mathbf{p}}_d^{k+1} - \mathbf{p}_d^o =& (\mathbf{I} - \beta \mathbf{\Gamma}\mathbf{\Sigma}\mathbf{\Gamma}^T)^k(\tilde{\mathbf{p}}_d^1 - \mathbf{p}_d^o) \\ =& (\mathbf{\Gamma}(\mathbf{I} - \beta \mathbf{\Sigma})\mathbf{\Gamma}^T)^k(\tilde{\mathbf{p}}_d^1 - \mathbf{p}_d^o) \end{aligned} \tag{4.65}$$

As ith coordinate on the right-hand side of (4.65) is multiplied with a power $(1 - \beta\varepsilon_i)^k$ From system stability point of view, if we denote ε_i as eigenvalue of Hessian matrix $\mathbf{H}_{\mathbf{p}_d^o}$

at optimum point, to prevent divergence, we need to have constraint on the learning rate [4]:

$$\beta < \frac{2}{\varepsilon_{max}} \tag{4.66}$$

Where ε_{max} is largest eigenvalue of Hessian matrix. To ensure convergence, the learning rate should fulfill condition [4]:

$$0 < \beta\varepsilon_i < 1 \tag{4.67}$$

Adapt-then-combine strategy
From observation of (4.54), we discover each individual cost function is different and therefore estimated parameters would converge to different values if we implement SGD individually for each receiver, as they are minimizing different cost functions. Thus, we need to find a way to search for optimal parameters in NMO model minimize global cost function in a distributed fashion and make estimated parameters $\tilde{u}_{d,f}$ and $\tilde{t}_{0,d,f}$ at different receiver in agreement after convergence.

$$\underset{\tilde{u}_{d,f},\tilde{t}_{0,d,f}}{\operatorname{argmin}} \; C_g(\tilde{u}_{d,f}, \tilde{t}_{0,d,f}) \tag{4.68}$$

As a consequence, the adapt-then-combine (ATC) strategy [8] is came up with to acheive this goal which incorporates a data exchange between local gradient of local cost function among neighbours and a data fusion stage that combines local estimate of parameters among neighbours with given weights. It consists of two-stages, at first we introduce an intermediate estimate of parameters that incorporates weighted combination of gradient of local cost function from neighbouring receivers corresponds to (4.69) and (4.71). This stage we called it adapt stage which aims to adapt and learn the error of local cost functions from these local gradients. At second stage, we combine local estimate of parameters corresponds to (4.70) and (4.72) This scheme enables an adaptive and continuous learning process towards global cost function and finally we would converge to global estimated parameters which minimizes global cost function. The detailed derivation of these update equation with ATC can be found in [8].

$$\bar{u}^{k+1}_{d,f} = \tilde{u}^{k}_{d,f} - \beta \sum_{i \in \mathcal{N}_d} W_{d,i} \frac{\partial c_d(\tilde{u}^k_{i,f}, \tilde{t}^k_{0,i,f})}{\partial \tilde{u}^k_{i,f}} \tag{4.69}$$

$$\tilde{u}^{k+1}_{d,f} = \sum_{i \in \mathcal{N}_d} W_{d,i} \bar{u}^{k+1}_{d,f} \tag{4.70}$$

Similarly for update of parameter $\tilde{t}^{k+1}_{0,d,f}$ we also have:

$$\bar{t}^{k+1}_{0,d,f} = \tilde{t}^{k}_{0,d,f} - \beta \sum_{i \in \mathcal{N}_d} W_{d,i} \frac{\partial c_i(\tilde{u}^k_{i,f}, \tilde{t}^k_{0,i,f})}{\partial \tilde{t}^k_{0,d,f}} \tag{4.71}$$

$$\tilde{t}_{0,d,f}^{k+1} = \sum_{i \in \mathcal{N}_d} W_{d,i} \tilde{t}_{0,d,f}^{k+1} \tag{4.72}$$

Convexity of cost function

The convexity of global cost function in (4.53) is crucial for set initial estimated parameters in ATC algorithm. As global cost function contains local cost function with same form, we only need to analyze convexity of local cost function. To check it, we come to calculate its 2nd order derivative as:

$$\begin{aligned} \frac{\partial^2 c_d(\tilde{u}_{d,f}, \tilde{t}_{0,d,f})}{\partial \tilde{u}_{d,f}^2} &= \frac{\partial (1 - \frac{\tilde{t}_{d,f}}{\sqrt{\tilde{u}_{d,f} {x_d}^2 + \tilde{t}_{0,d,f}^2}}) x_d^2}{\partial \tilde{u}_{d,f}} \\ &= \frac{1}{2} \frac{t_{d,f} x_d^4}{(\tilde{u}_{d,f} {x_d}^2 + \tilde{t}_{0,d,f}^2)^{\frac{3}{2}}} \end{aligned} \tag{4.73}$$

From observing this second order derivative, we discover it's positive and the denominator term is comparatively very small than numerator, therefore the second order derivative with respect to slowness is very large. Also, for second order derivative with respect to parameter $\tilde{t}_{0,d,f}$:

$$\begin{aligned} \frac{\partial^2 c_d(\tilde{u}_{d,f}, \tilde{t}_{0,d,f})}{\partial \tilde{t}_{0,d,f}^2} &= \frac{\partial 2(\tilde{t}_{0,d,f}(1 - \frac{t_{d,f}}{\sqrt{\tilde{u}_{d,f} {x_d}^2 + \tilde{t}_{0,d,f}^2}}))}{\partial \tilde{t}_{0,d,f}} \\ &= 2(1 - \frac{t_{d,f}}{\sqrt{\tilde{u}_{d,f} {x_d}^2 + \tilde{t}_{0,d,f}^2}} + \frac{\tilde{t}_{0,d,f}^2 t_{d,f}}{(\tilde{u}_{d,f} {x_d}^2 + \tilde{t}_{0,d,f}^2)^{\frac{3}{2}}}) \end{aligned} \tag{4.74}$$

To analyze convexity of local cost function, we also come to Hessian matrix associated with these second order derivatives as:

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 c_d(\tilde{u}_{d,f}, \tilde{t}_{0,d,f})}{\partial \tilde{u}_{d,f}^2} & \frac{\partial c_d(\tilde{u}_{d,f}, \tilde{t}_{0,d,f})}{\partial \tilde{u}_{d,f} \tilde{t}_{0,d,f}} \\ \\ \frac{\partial c_d(\tilde{u}_{d,f}, \tilde{t}_{0,d,f})}{\partial \tilde{t}_{0,d,f} \tilde{u}_{d,f}} & \frac{\partial^2 c_d(\tilde{u}_{d,f}, \tilde{t}_{0,d,f})}{\partial \tilde{t}_{0,d,f}^2} \end{bmatrix} \tag{4.75}$$

Also, the second order derivatives $\frac{\partial c_d(\tilde{u}_{d,f},\tilde{t}_{0,d,f})}{\partial \tilde{u}_{d,f}\tilde{t}_{0,d,f}}$ and $\frac{\partial c_d(\tilde{u}_{n,j},\tilde{t}_{0,d,f})}{\partial \tilde{t}_{0,d,f}\tilde{u}_{d,f}}$ is calculated as:

$$\begin{aligned}\frac{\partial c_d(\tilde{u}_{d,f},\tilde{t}_{0,d,f})}{\partial \tilde{u}_{d,f}\tilde{t}_{0,d,f}} &= \frac{\partial(1-\frac{\tilde{t}_{d,f}}{\sqrt{\tilde{u}_{d,f}x_d{}^2+\tilde{t}^2_{0,d,f}}})x_d^2}{\partial \tilde{t}_{0,d,f}} \\ &= \frac{\partial(\frac{t_{d,f}}{\sqrt{\tilde{u}_{d,f}x_d{}^2+\tilde{t}^2_{0,d,f}}})x_d^2}{\partial \tilde{t}_{0,d,f}} \\ &= \frac{t_{d,f}x_d^2\tilde{t}_{0,d,f}}{(\tilde{u}_{d,f}x_d{}^2+\tilde{t}^2_{0,d,f}))^{\frac{3}{2}}}\end{aligned} \tag{4.76}$$

$$\frac{\partial c_d(\tilde{u}_{d,f},\tilde{t}_{0,d,f})}{\partial \tilde{t}_{0,d,f}\tilde{u}_{d,f}} = \frac{t_{d,f}x_d^2\tilde{t}_{0,d,f}}{(\tilde{u}_{d,f}x_d{}^2+\tilde{t}^2_{0,d,f}))^{\frac{3}{2}}} \tag{4.77}$$

If cost function is strictly convex, the determinant of Hessian matrix should be positive, which serves a necessary condition for convexity:

$$det(\mathbf{H}) > 0 \tag{4.78}$$

After calculation of Hessian matrix we would finally yield a lower bound on receiver position:

$$x_d > \sqrt{\frac{t^2_{d,f}-\tilde{t}^2_{0,d,f}}{\tilde{u}_{d,f}}} \tag{4.79}$$

With constraint $t_{d,f} >= \tilde{t}_{0,d,f}$ and $\tilde{u}_{d,f}$ is positive. Detailed derivation of (4.79) can be found in Appendix. Generally the magnitude of optimal estimated slowness $\tilde{u}^o_{d,f}$ is much smaller compared with optimal estimated $\tilde{t}^o_{0,d,f}$ and picking time $t_{d,f}$. Assume (4.79) holds on during estimation process and then estimated parameters would finally converge to optimal parameters. This means receiver distance to source should be varied and would be very large to fulfill (4.79) and ensure strict convexity and certainly beyond maximum length of our simulation scenario. Therefore, cost function is generally non-convex for our simulation scenario.

Algorithm structure

With above analysis, non-convexity of cost function makes a good initial guess of parameters is necessary for ATC scheme to obtain an acceptable performance. Then, we could formulate a distributed NMO algorithm with ATC given in algorithm 6, which is explained as follows: the initialization of distributed NMO with adapt-then-combine is same as that with average consensus. Firstly, we choose a certain picking algorithm to get picking travel time of reflection wave for each receiver measurement. Then each receiver performs a nonlinear least-square fitting based on picking travel time from their neighbors to get initial estimated parameters at receivers, corresponds to line 1-13. Then we enter into inner loop of algorithm, firstly perform an estimation of layer velocity and depth with NMO with current estimated parameters at each iteration corresponds to line 17-20. Then we share gradient of local cost function at neighbors to get intermediate estimated parameters and combine them with second averaging process over these intermeidate estimated parameters to get updated estimated parameters at next iteration. This corresponds to line 21-26. To the end of algorithm, we would obtain same estimated layer velocity and thickness for each layer after all receivers converge.

Estimation performance analysis

To simulate how estimation performs under the distributed NMO algorithm with adapt-then-combine. We apply simulation set up shown in Table 4.2 and also use line topology for neighbor arrangement.

Table 4.2: Simulation setup for distributed NMO with ATC

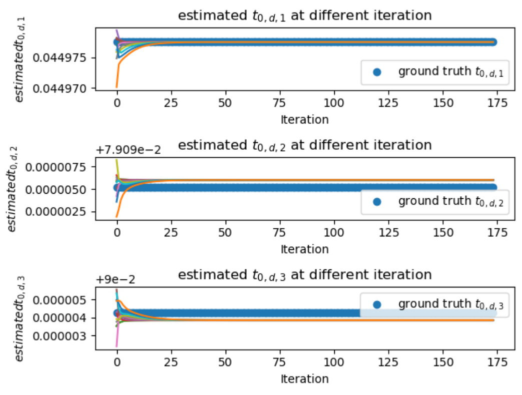
length X	**number of receivers**	**number of neighbors** L_d	**SNR**	**learning rate** β	**iteration** k_{max}
28 m	24	4	5dB	4e-12	174

The estimated parameters temporal variation at different receiver are plotted in Fig 4.15. Different solid lines with different color represent estimated parameter curve at different iterations at different receiver, while blue dots reprsent correponding ground truth. Clearly, initially each receiver has different estimated parameter but finally they would consensus on a certain parameter with adapt-then-combine algorithm.

The velocity and depth estimation performance at different iterations are plotted in Fig 4.15 and Fig 4.16. Initially, we could find estimated layer velocity and depth show similar pattern as in the previous average consensus case in Fig 4.9 (a) and Fig 4.10 (a). This is because we get same picking in both distributed schemes that turns out same initial estimated parameters at receiver. However, we firstly exchange initial parameters and perform estimation for distributed NMO with average consensus, but in distributed NMO with ATC, we perform estimation firstly with initial parameters and then do data exchange. This results in difference between initial estimation pattern of two schemes, but their general pattern is very similar. Then, from Fig 4.15 (b)-(d) and Fig 4.16 (b)-(d), we observe ATC update drives parameters update to minimize global cost function and therefore we observe estimation performance is gradually becoming better and we observe convergence of all estimated parameters at all receivers at 136th iteration.

Algorithm 6 Distributed NMO with adapt-then-combine

```
 1: for $\mathbf{z}_d^n$ d=1,...,D: do
 2:     choose picking method
 3:     if picking from deconvolved measurement then
 4:         $\hat{\boldsymbol{\mu}}_d = (\frac{\mathbf{W}^T\mathbf{W}}{\sigma^2} + \lambda\mathbf{I})^{-1}\frac{\mathbf{W}^T\mathbf{z}_d^n}{\sigma^2}, \lambda = \frac{1}{\sigma^3}$
 5:         travel-time picking
 6:     else
 7:         picking from STA/LTA picker
        end if
 8:     $\mathbf{t} = [\mathbf{t}_1, \mathbf{t}_2, \dots, \mathbf{t}_{f_r}]^T$
    end for
 9: for receiver $d = 1, \dots, D$: do
10:     gather arrival time estimate and receiver offset from its $L_d$ neighbours:
11:     $\mathbf{t}_d^N = [\mathbf{t}_{d,1}^N, \mathbf{t}_{d,2}^N, \dots, \mathbf{t}_{d,L_d}^N]^T$
12:     $\mathbf{x}_d^N = [x_{d,1}^N, x_{d,2}^N, \dots, x_{d,L_d}^N]^T$
13:     Nonlinear least-square fitting: $\to \hat{t}^1_{0,d,f}, \hat{u}^1_{d,f}, f = 1, \dots, f_r$
    end for
14: for iteration $k = 1, 2, \dots, k_{max}$: do
15:     for layer $f = 1, \dots, f_r$: do
16:         for receiver $d = 1, \dots, D$: do
17:             estimate velocity and depth with local estimated parameters:
18:             $\tilde{v}^{rms-k}_{d,f} = \frac{1}{\sqrt{\tilde{u}^k_{d,f}}}$
19:             $\to \tilde{v}^k_{d,f} = \sqrt{\frac{\tilde{v}^{rms-k2}_{d,f}\tilde{t}^k_{0,d,f} - \tilde{v}^{rms-k2}_{d,f-1}\tilde{t}^k_{0,d,f-1}}{\tilde{t}^k_{0,d,f} - \tilde{t}^k_{0,d,f-1}}}$
20:             $\to \tilde{h}^k_{d,f} = \frac{1}{2}(\tilde{t}^k_{0,d,f} - \tilde{t}^k_{0,d,f-1})\tilde{v}^k_{d,f}$
21:             Adapt stage:
22:             $\bar{m}^{k+1}_{d,f} = \tilde{u}^k_{d,f} - \beta\sum_{i\in\mathcal{N}_d} W_{d,i}\frac{\partial c_d(\tilde{u}^k_{d,i}, \tilde{t}^k_{0,i,f})}{\partial \tilde{u}^k_{i,f}}$
23:             $\bar{t}^{k+1}_{0,d,f} = \tilde{t}^k_{0,d,f} - \beta\sum_{i\in\mathcal{N}_d} W_{d,i}\frac{\partial c_i(\tilde{u}^k_{i,f}, \tilde{t}^k_{0,i,f})}{\partial \tilde{t}^k_{0,d,f}}$
24:             Combine stage:
25:             $\tilde{u}^{k+1}_{d,f} = \sum_{i\in\mathcal{N}_d} W_{d,i}\bar{u}^{k+1}_{d,f}$
26:             $\tilde{t}^{k+1}_{0,d,f} = \sum_{i\in\mathcal{N}_d} W_{d,i}\bar{t}^{k+1}_{0,d,f}$
            end for
        end for
    end for$\to \tilde{t}^{k_{max}}_{0,d,f}, \tilde{u}^{k_{max}}_{d,f}, \tilde{h}^{k_{max}}_{d,f}, \tilde{v}^{k_{max}}_{d,f}$
```


(a) estimated $t_{0,d,f}$

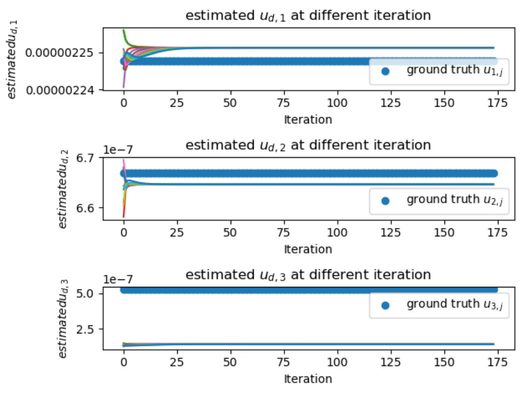
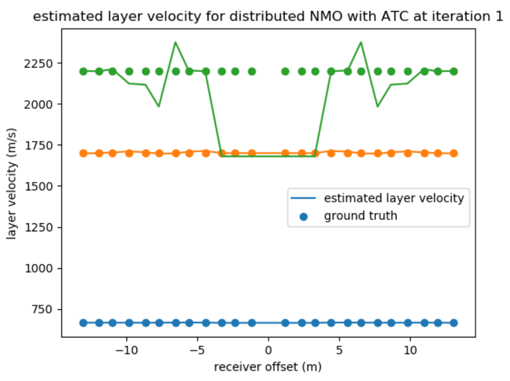
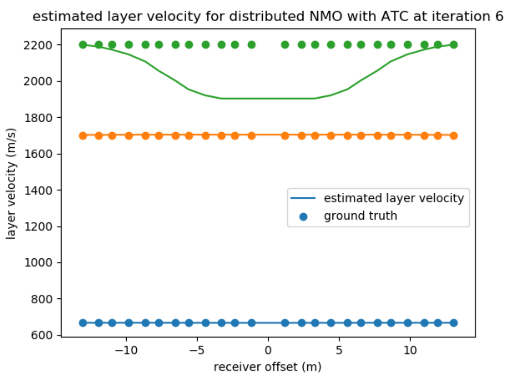
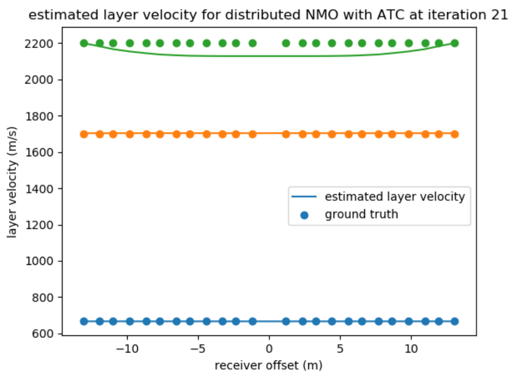

(b) estimated $u_{d,f}$

Figure 4.14: Estimated parameters variation of distributed NMO with ATC


(a) initial estimation


(b) 6th iteration


(c) 21th iteration

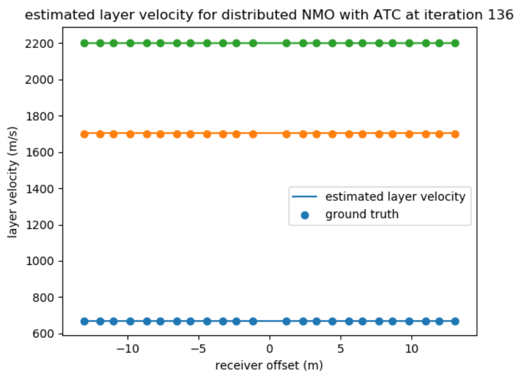
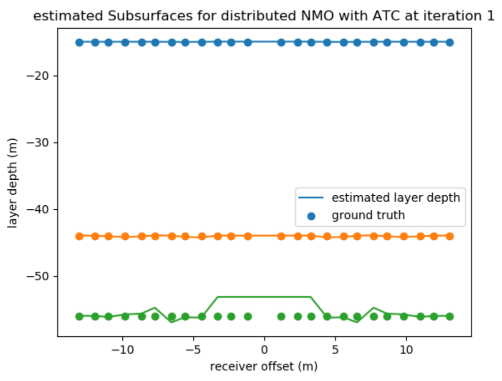
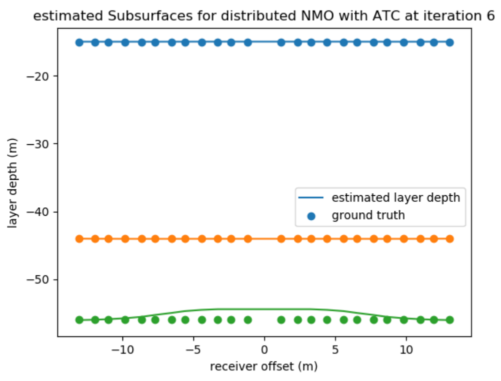

(d) 136th iteration

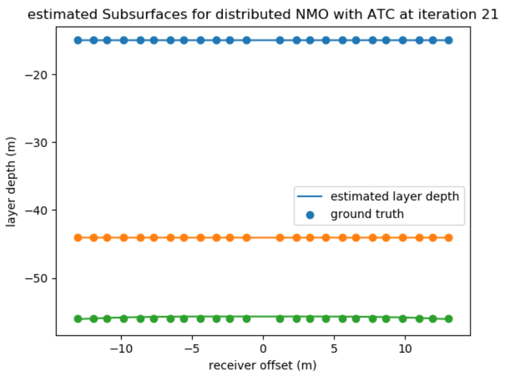
Figure 4.15: Velocity estimation results for distributed NMO with ATC



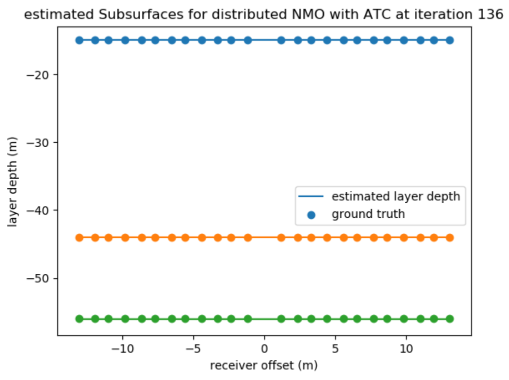
(a) initial estimation



(b) 6th iteration



(c) 21th iteration



(d) 136th iteration

Figure 4.16: Depth estimation results for distributed NMO with ATC

4.3 Comparing analysis of NMO algorithms

In this section, a series of comparative studies about NMO algorithms are performed and analyzed, in terms of estimation performance with different neighbor arrangement, different NMO algorithms with noiseless and noisy link.

4.3.1 Comparison of neighbor arrangement topology

An interesting question is how the constellation of neighbors influences estimation performance of distributed NMO algorithm. Therefore, we take distributed NMO algorithm with average consensus as an instance and apply two different topologies to see how it affects estimation performance: **random topology** and **line toplogy**. The simulation setup is same as given in Table 4.2 and also we use same picking that get from deconvolved profile in the previous section.

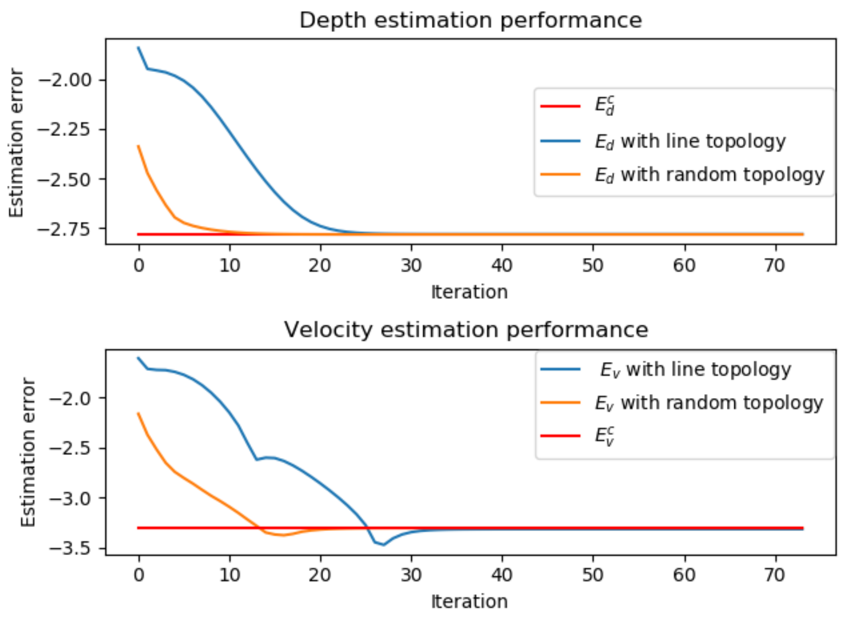
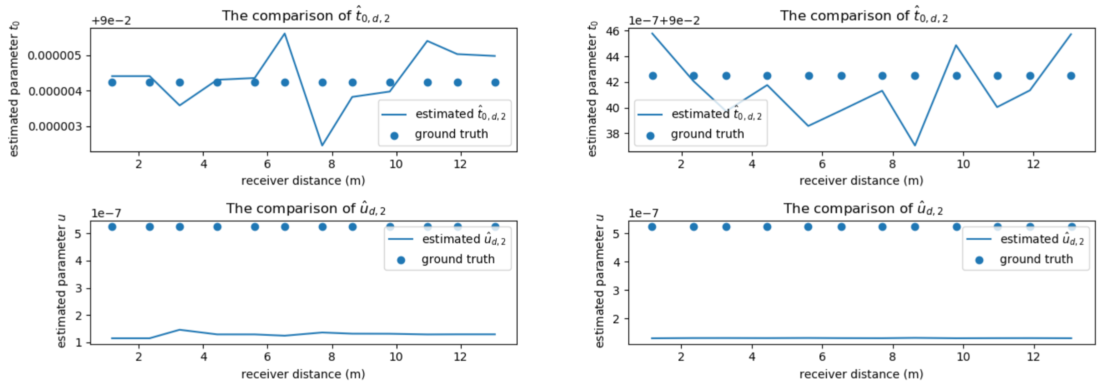


Figure 4.17: Estimation performance comparison with different neighbor arrangement

In Fig 4.17, we plot initial estimated parameters at different receivers on one side for both random and line topologies. From Fig 4.17, we discover estimation error converges faster with random topology, instead of line topology. This is because in line topology, receiver closer to source would collect picking travel time which are closer to each other, which turns out estimated parameters would generally deviate more with larger source-to-receiver distance. This can be seen in Fig 4.18 (a). However, for random topology



(a) line topology (b) random topology

Figure 4.18: Initial estimated parameters at different receivers for different topology

estimated parameters show a random pattern with smaller variance compared with line topology, which can been by looking at y-axis of Fig 4.18 (a)-(b). Thus, initial estimated parameters at different receivers are relatively closer to their average and would achieve consensus faster in the random topology.

4.3.2 Estimation performance comparison with noiseless links

Estimation performance comparison between two distributed NMO algorithms is important to choose correspond algorithm that can achieve better performance in same simulation scenario.

Therefore, in Fig 4.19 we plot estimation error curve comparison of these two distributed NMO and centralized NMO algorithms. In both schemes, we use same picking method with deconvolved measurement and apply line topology for neighbor arrangement and simulation set up is same as previous in table 4.1 and table 4.2. Additionally, the velocity and depth estimation error of distributed NMO with ATC are defined in same logarithmic form as E_v^{ATC} and E_d^{ATC}:

$$E_v^{ATC} = \log\left(\frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\tilde{v}_{d,f}-v_{d,f}|}{|v_{d,f}|}}{Df_r}\right) \tag{4.80}$$

$$E_d^{ATC} = \log\left(\frac{\sum_{d=1}^{D}\sum_{f=1}^{f_r}\frac{|\tilde{h}_{d,f}-h_{d,f}|}{|h_{d,f}|}}{Df_r}\right) \tag{4.81}$$

Obviously, distributed NMO with adapt-then-combine converges slower than distributed

NMO with average consensus. The convergence rate of ATC framework is closely related to learning rate. The smaller learning rate within the range that ensure convergence, updated parameters move slower towards minimum point and thus cause slower convergence. Thus, the velocity estimation error converge much slowly at around 110 iteration because we choose a very small learning rate of 4e-12. Interestingly, distributed NMO with ATC shows better performance in steady state shown in orange solid line than that of average consensus shown in blue solid line. Because it involves two optimization procedures regarding to estimated parameter. First one is getting estimated parameters with nonlinear least-square fitting, proceed with diffusion learning that search optimal parameters minimize global cost function. Therefore, we would finally achieve very good result with good starting point from accurate picking with deconvolved measurement. However, in dsitributed NMO with average consensus we only perform fitting and then average over these estimated parameters. This may cause a larger bias in final estimated layer velocity and depth than distributed NMO with ATC.

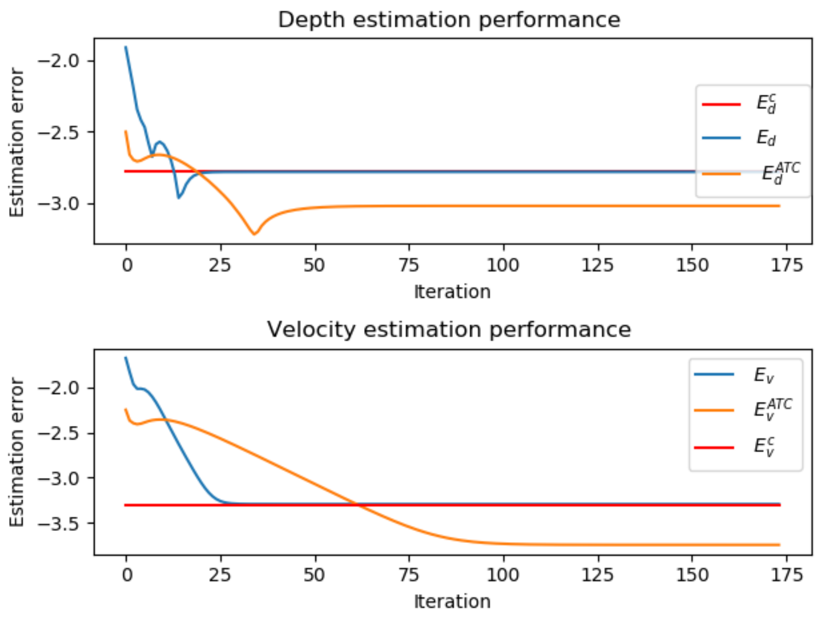


Figure 4.19: Estimation performance comparison with picking from deconvolved measurement

More importantly, estimation performance comparison regarding to an inaccurate picking should also be performed to see how both schemes deal with worse initialization conditon. Thus, we apply STA/LTA picker with same simulation setup in previous comparison scenario and plot velocity estimation performance in Fig 4.20 and Fig 4.21. By comparing both figures, both distributed schemes show very worse initial performance due to inaccurate picking from STA/LTA picker which cause bad initial estimated parameters from fitting. But with algorithm update, distributed NMO with average consensus still reach a very good estimation result at 37th iteration shown in Fig 4.20 (d). In contrast, distributed NMO with ATC shows an apparent bias after consensus in the estimated layer velocity at 136th iteration shown in Fig 4.21 (d). This is because we perform correction of picking at each step in distributed NMO with average consensus. Then estimated root mean square velocity could be written as:

$$\begin{aligned}\hat{v}_{d,f}^{rms-k_{max}} =& \frac{x_d}{\sqrt{\hat{t}_{d,f}^2 - \hat{t}_{0,d,f}^{k_{max}}}} \\ =& \frac{x_d}{\sqrt{\hat{u}_{d,f}^{k_{max}}}}\end{aligned} \tag{4.82}$$

From average consensus point of view, $\hat{u}_{d,f}^{k_{max}}$ would be average of all initial estimate as:

$$\hat{u}_{d,f}^{k_{max}} = \frac{1}{D}\sum_{d=1}^{D}\hat{u}_{d,f}^1 = \frac{1}{D}\sum_{d=1}^{D}\Delta\hat{u}_{d,f}^1 + u_{d,f} \tag{4.83}$$

The right hand side of (4.83) is done by replacing initial estimate with ground truth and a bias term:

$$\hat{u}_{d,f}^1 = u_{d,f} + \Delta\hat{u}_{d,f}^1 \tag{4.84}$$

As initial estimated slowness $\hat{u}_{d,f}^1$ and two way vertical travel time $\hat{t}_{0,d,f}^1$ are damping around ground truth and thus bias term could be negative or positive at different receivers, which can be seen in Fig 4.21. This turns out the averaged bias term $\frac{1}{D}\sum_{d=1}^{D}\Delta\hat{u}_{d,f}^1$ is very small after summing over local bias at different receivers. Thus, we still achieve very good result. But in distributed NMO with ATC, worse picking will lead to a corresponding worse initial estimated slowness. This also points out advantage of distributed NMO with average consensus that it is not required that each receiver has very good initial estimated parameter, as these estimated parameters could be damping around ground truth and finally also gives us a small averaged bias term that leads to good ultimate estimation performance. But in distributed NMO with ATC, due to non-convexity of cost function, it would converge to its near local optimum parameter or saddle point if initial estimated parameters are not sufficiently good, which is reflected by the large bias in estimation performance after convergence. From these scheme comparisons, we could find the superiority of distributed NMO with average consensus to deal with rather inaccurate picking. But with sufficiently accurate picking, distributed NMO with ATC could perform better with diffusion learning over parameters.

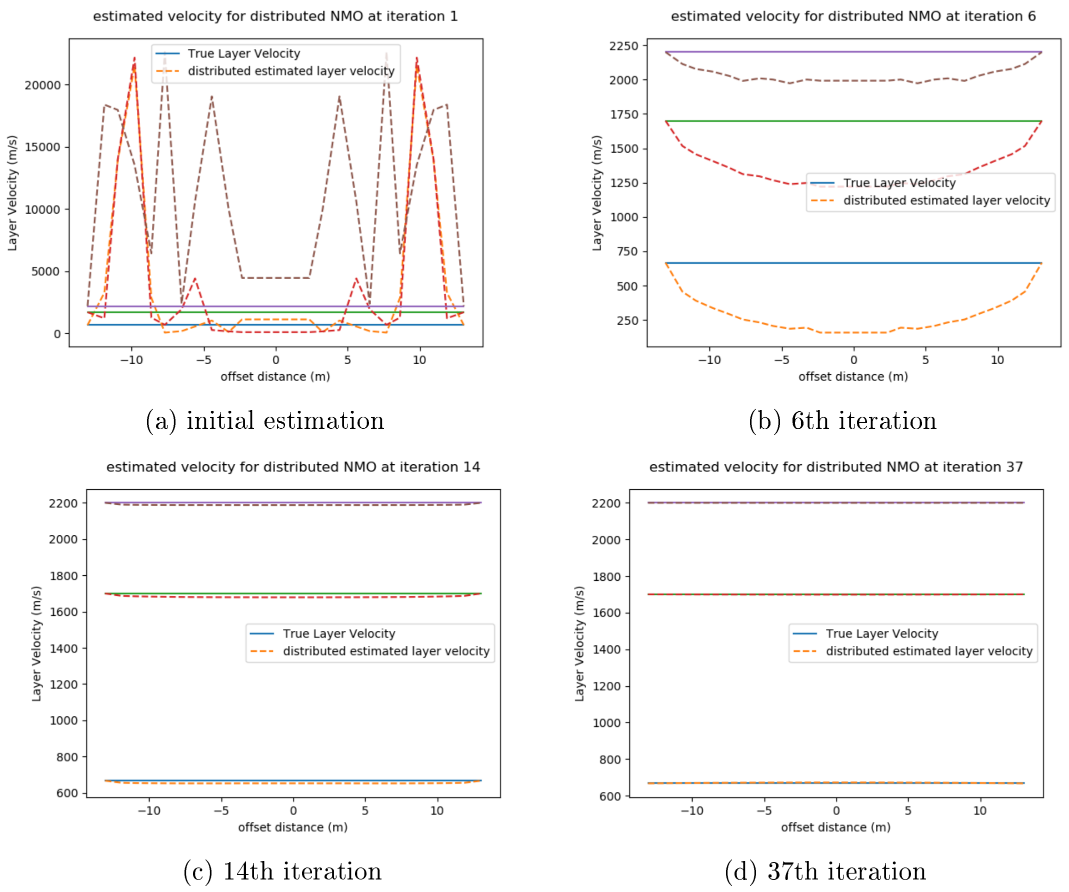
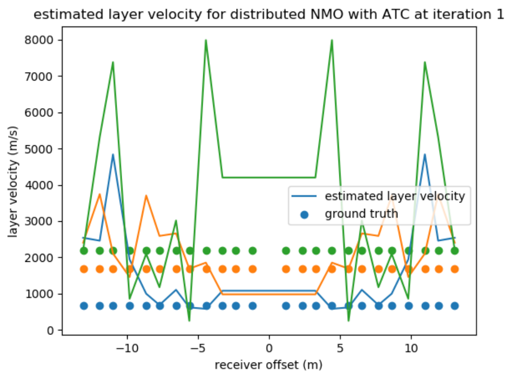
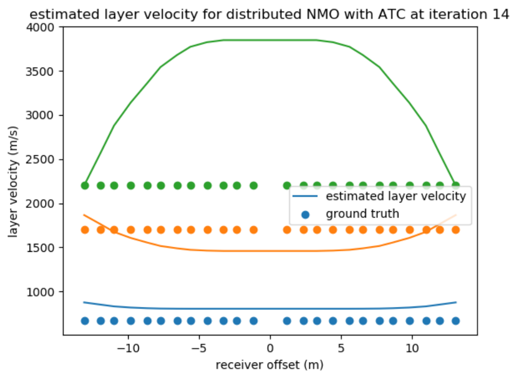


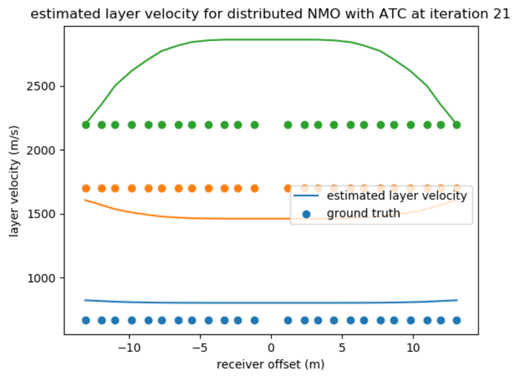
Figure 4.20: Velocity estimation results for distributed NMO with average consensus using STA/LTA picker



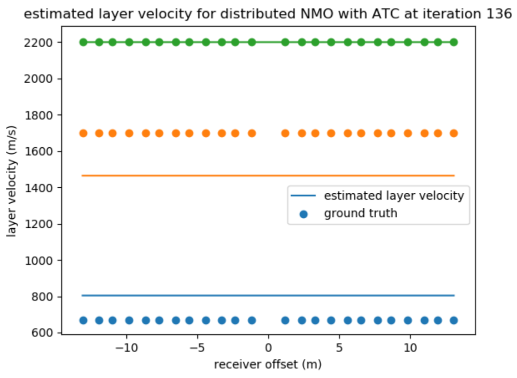
(a) initial estimation



(b) 14th iteration



(c) 21th iteration



(d) 136th iteration

Figure 4.21: Velocity estimation results for distributed NMO with ATC using STA/LTA picker

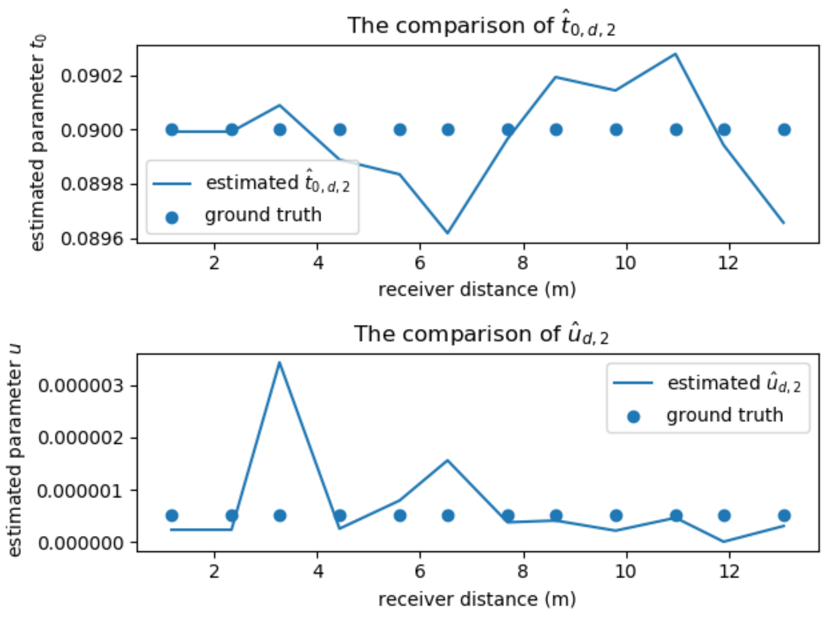


Figure 4.22: initial estimated parameters at different receiver with STA/LTA picker

4.3.3 Comparing analysis with noisy inter-receiver links

In previous algoithm design of distributed normal moveout, the addtive noise during data exchange and update between receiver and its neighbours is neglected. However, physical data transmission channel is always noisy and thus updated parameters collecting from neighbors would always be noisy. Firstly, we analyze how estimated parameters varies with noisy inter-receiver link with distributed NMO with average consensus.

Distributed NMO with average consensus with noisy inter-receiver link

We take estimated two-way vertical travel time $\hat{\mathbf{t}}_{0,f}^{k+1}$ as an example to illustrate update parameter variation with noisy link. The average consensus update of $\hat{\mathbf{t}}_{0,f}^{k+1}$ in the whole receiver network for layer f is described by (4.85).

$$\hat{\mathbf{t}}_{0,f}^{k+1} = \mathbf{W}\hat{\mathbf{t}}_{0,f}^{k} + \mathbf{n}^k \tag{4.85}$$

Where $\mathbf{n}^k \sim \mathcal{N}(0, \sigma^2)$. When data is transfered with noisy link, it becomes a stochastic process. In order to investigate how data exchange process varies with additive noise via link in distributed normal moveout. We can directly come to analyze how the mean of update estimated parameters varies in a noisy link:

$$\begin{aligned}\mathcal{E}(\hat{\mathbf{t}_{0,f}^{k}}) =& \mathcal{E}(\mathbf{W}\hat{\mathbf{t}_{0,f}^{k-1}}) + \mathcal{E}(\mathbf{n}^{k-1}) \\ =& \mathcal{E}(\mathbf{W}^2\hat{\mathbf{t}_{0,f}^{k-2}}) + \mathcal{E}(\mathbf{W}\mathbf{n}^{k-2}) + \mathcal{E}(\mathbf{n}^{k-1}) \\ =& \underbrace{\mathcal{E}(\mathbf{W}^k\hat{\mathbf{t}_{0,f}^{0}})}_{average} + \mathcal{E}(\mathbf{W}\mathbf{n}^{k-2}) + \cdots + \underbrace{\mathcal{E}(\mathbf{n}^{k-1})}_{0}\end{aligned} \tag{4.86}$$

From (4.86) we could discover with additive noise we introduce more and more disturbance to mean of initial estimated $\mathbf{t}_0$ with increasing time step in the network. Therefore we would not observe convergence of distributed consensus average with noisy inter-receiver link. To observe updated parameters variation with algorithm update, we perform simulation with same setup in 4.1 and SNR of noisy link is set as 30dB. The local estimated two way vertical travel time at different receivers for different layers $\hat{t}_{0,d,f}$ show a divergent trend with noisy link and receivers would not achieve consensus and deviate from initial average with increasing time as general trend according to (4.86), which is clearly shown in Fig 4.23.

Additionally, we could also perform analysis towards variation of average of estimated parameters at all receiver. Firstly, we define an average vector of initial receiver information with total D receivers as:

$$\boldsymbol{\mathcal{T}}_{0,f}^{k+1} = \frac{1}{D}\mathbf{l}\mathbf{l}^T\hat{\mathbf{t}}_{0,f}^{k+1} \tag{4.87}$$

Then multiply both sides of (4.85) with $\frac{1}{D}\mathbf{l}\mathbf{l}^T$:

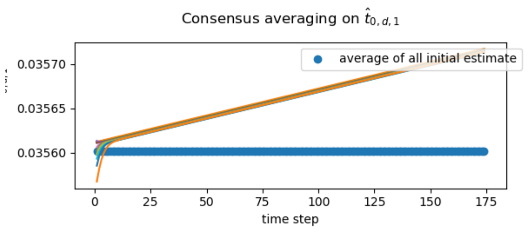
$$\boldsymbol{\mathcal{T}}_{0,f}^{k+1} = \boldsymbol{\mathcal{T}}_{0,f}^{k} + \frac{1}{D}\mathbf{l}\mathbf{l}^T\mathbf{n}^k \tag{4.88}$$

The expectation can be calculated as:

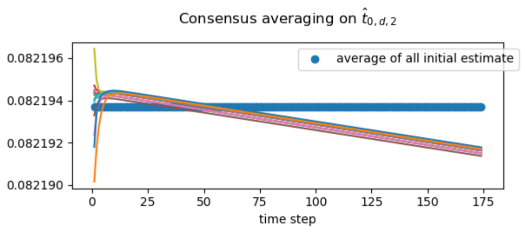
$$\mathcal{E}(\boldsymbol{\mathcal{T}}_{0,f}^{k}) = \boldsymbol{\mathcal{T}}_{0,f}^{0} \tag{4.89}$$

Which is constant, but the expectation of variance is different as:

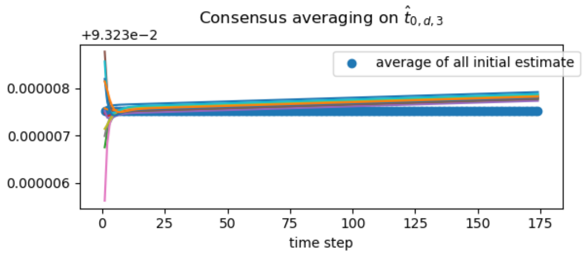
$$\begin{aligned}\mathcal{E}((\boldsymbol{\mathcal{T}}_{0,f}^{k})^2) &= \mathcal{E}(\boldsymbol{\mathcal{T}}_{0,f}^{k} - \mathcal{E}(\boldsymbol{\mathcal{T}}_{0,f}^{k}))^2 \\ &= \mathcal{E}(\mathcal{T}_{0,f}^{k} - \boldsymbol{\mathcal{T}}_{0,f}^{1})^2 \\ &= \mathcal{E}((\boldsymbol{\mathcal{T}}_{0,f}^{k})^2) + \mathcal{E}((\boldsymbol{\mathcal{T}}_{0,f}^{1})^2) - 2\mathcal{E}(\boldsymbol{\mathcal{T}}_{0,f}^{k}\boldsymbol{\mathcal{T}}_{0,f}^{0}) \\ &= var(\boldsymbol{\mathcal{T}}_{0,f}^{k}) \\ &= var(\boldsymbol{\mathcal{T}}_{0,f}^{1} + \sum_{i=1}^{k}\mathbf{n}^i) \\ &= k\sigma^2\end{aligned} \tag{4.90}$$



(a) $\hat{t}_{0,d,1}$ variation



(b) $\hat{t}_{0,d,2}$ variation



(c) $\hat{t}_{0,d,3}$ variation

Figure 4.23: $\hat{t}_{0,d,f}$ at different receiver for distributed NMO with average conensus with inter-receiver noisy link

This means AWGN introduce an error in the average node value which has a linearly increasing variance with time step.

Distributed NMO with ATC with noisy inter-receiver link

In the distributed NMO with ATC algorithm, additive noise is added when each receiver shares its local gradient of local cost function to neighbor and also at combine stage that each receiver collects intermediate estimated parameters from neighbors. Recall (4.71)-(4.72), the updated estimated parameters at receivers could be described as matrix form in (4.91):

$$\begin{aligned}\tilde{\mathbf{t}}_{0,f}^{k+1} &=\mathbf{W}\bar{\mathbf{t}}_{0,f}^{k+1} + \mathbf{n}_c^k \\ &=\mathbf{W}(\tilde{\mathbf{t}}_{0,f}^k - \beta\mathbf{W}\mathcal{H}^k - \mathbf{n}_a^k) + \mathbf{n}_c^k \\ &=\mathbf{W}(\tilde{\mathbf{t}}_{0,f}^k - \beta\mathbf{W}\mathcal{H}^k) + \mathbf{n}_c^k - \mathbf{W}\mathbf{n}_a^k\end{aligned} \tag{4.91}$$

Where $\mathbf{n}_c^k$ and $\mathbf{n}_a^k$ are AWGN added in the date transmission of combine and adapt stage and $\mathcal{H}^k = [\frac{\partial c_1(\tilde{u}_{1,f}^k,\tilde{t}_{0,1,f}^k)}{\partial \tilde{t}_{0,1,f}^k}, \frac{\partial c_2(\tilde{u}_{2,f}^k,\tilde{t}_{0,2,f}^k)}{\partial \tilde{t}_{0,2,f}^k}, \ldots, \frac{\partial c_D(\tilde{u}_{D,f}^k,\tilde{t}_{0,D,f}^k)}{\partial \tilde{t}_{0,D,f}^k}]^T$ is local gradient vector of cost function at all receivers. From (4.91), we discover noisy inter-receiver link introduces not only a AWGN term, but also a colored gaussian noise term with recursive update of distributed NMO with ATC algorithm.

This is differing from the case for distributed NMO with average consensus, where we only observe an AWGN term added in the update equation in (4.85). Also, consensus of estimated parameters cannot achieve under noisy links, as each step we add some noise in the updated parameters. Ultimately, we could perform comparative investigation about estimation performance regarding to noisy inter-receiver link with same SNR in the communication channel. The simulation setup for both distributed NMO schemes are given in Table 4.1 and Table 4.2. For picking method, we apply picking from deconvoled profile. The SNR in the noisy inter-receiver link is set as 30dB. The comparison of estimation error between two distributed schemes are plotted in Fig 4.25, where distributed NMO with ATC shows worse performance than distributed NMO with average consensus under same SNR of noisy link. Because in ATC algorithm, we perform data exchange twice to update parameters which means noise effect is added up twice, but in distributed NMO with average consensus noise would only impact on updated parameters only once. Also in Fig 4.24 estimated $t_{0,d,f}$ in distributed NMO with ATC shows an general trend that deviating from ground truth with increasing iterations due to more and more noise added in the updated estimated parameters.

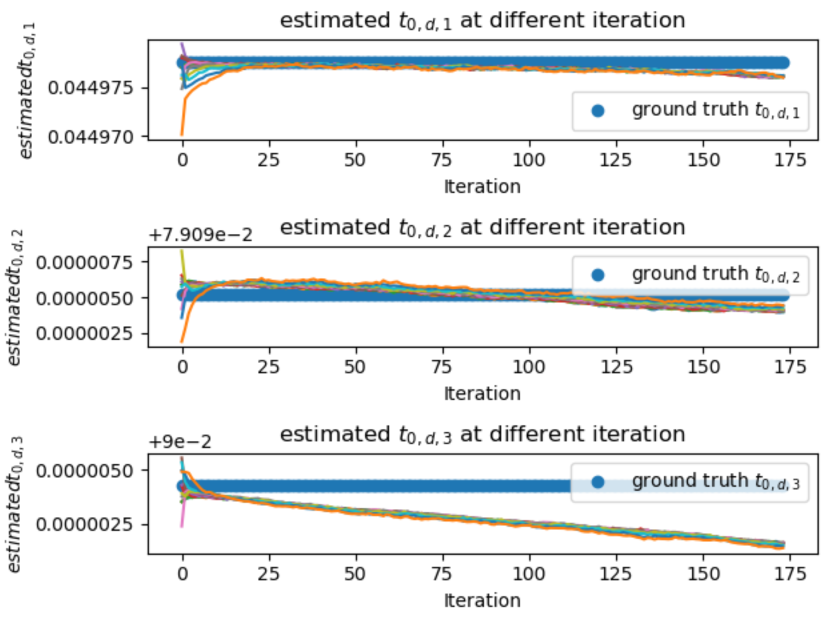


Figure 4.24: estimated $t_{0,d,f}$ for noisy link at different iterations in distributed NMO with ATC

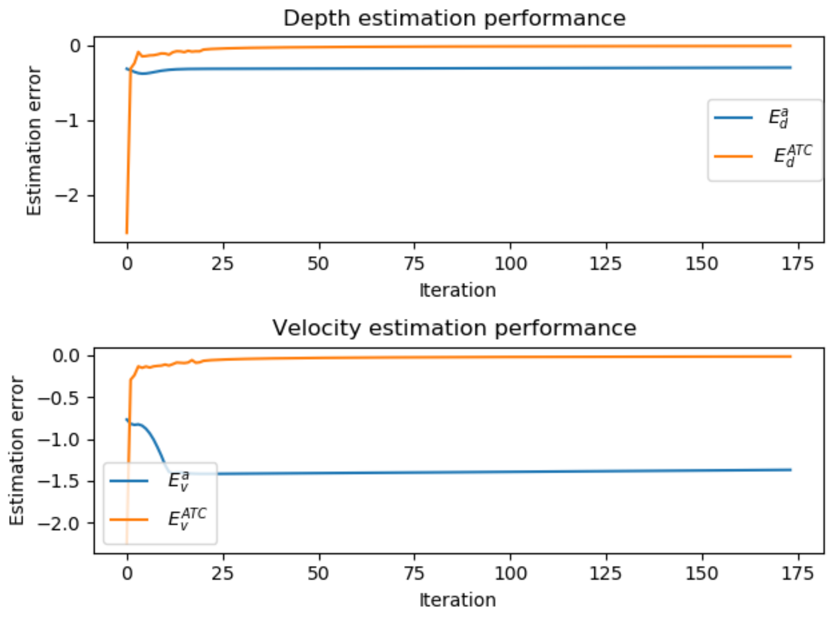


Figure 4.25: Estimation performance comparison between different distributed schemes

Chapter 5

Conclusion and future work

In this thesis, the concept and derivation of 2D wave equation are illustrated and its numerical solution of FDTD method that enables us to visualize wave propagation pattern in a subsurface model consisting of 3 horizontal layers with given wave propagation velocity in each layer by injecting a Ricker wavelet at ground source location. Afterwards, the noisy synthetic seismic measurement is recorded by receivers which are symmetrically positioned around source. To pick travel time of reflection wave arrives at receivers, we provide two approaches: picking from deconvolved seismic measurement and STA/LTA picker. The picking from deconvolved profile is more accurate because we add a picking correction stage.

The centralized NMO and two novel distributed NMO algorithms: distributed NMO with average consensus and distributed NMO with ATC are developed to estimate layer velocity and depth of aforementioned subsurface model based on picking travel time and analytical NMO model. The distributed algorithms realize the autonomous and coorperative estimation locally at different receivers by exchanging and achieve consensus on estimated parameters between receivers and their neighbors in noise-free channel, which is much more flexible and time-saving compared with centralized one. In noisy inter-receiver link, their estimation performance would not converge to certain value, as consensus of parameters cannot hold on with AWGN. The distributed NMO with average consensus shows better performance than that of ATC framwork in the case of inaccurate picking, but a worse performance for the case of sufficiently precise picking. As non-convexity of cost function makes a sufficient good starting point is necessary for ATC algorithm. Thus, inaccurate picking will result in bad initial estimate and may cause estimated parameters stuck at local optimum. Additionally, more neighbors acclerate averaging consensus procedure and achive faster convergence of distributed NMO with average consensus algorithm. The comparative studies towards different neighbor arrangement are also performed, which reveals a line topology will make distributed NMO with average consensus converges slower than that of random topology.

For the future work, the automatic picking algorithm for seismic measurement should be developed that enables us to directly obtain picking without knowledge of ground truth arriavl time, because picking travel time in this thesis relies on ground truth arrival time of reflection wave, which doesn't exist in reality. Also, as we only consider simple horizontal subsurface structures, which is not the case in reality with unevenly distributed reflective interface. The anisotropicity of layer will also cause velocity variation within one layer. All those factors are out of consideration in this thesis. Thus, for future work with complicated layer structures , full waveform inversion may be considered that finds optimal estimated velocity field of given subsurface model via comparing observed waveform with waveform from synthesized model. Then based on estimated velocity field, it is feasible to calculate travel time of seismic ray arrive at receivers by solving Eikonal equation and inversely reconstruct depth model with these calcualted travel time.

Chapter 6

Appendix

Derivation of 2D wave equation

The derivation of wave equation starts with introducing a homogenous and elastic geological material. Firstly, we consider a small area dA in this material with weight m defined by dx and dy and assume tension T is same everywhere in this material and deflection of material due to wave $\mathbf{a}(x, y, t)$ is very small. From Stress-strain relationship, when the incoming wave hits this material, it will cause deformation on left and right-hand side shown in Fig 6.1.

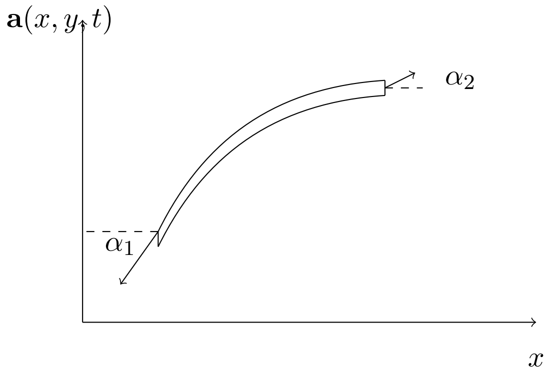


Figure 6.1: A tensioned geological material

The tension of leftmost and rightmost side of this material is same and equal to Tdy Then total acting force in horizontal direction on left and right-hand side is:

$$\mathbf{F}_h^1 = Tdy \cos\alpha_2 - Tdy \cos\alpha_1 \tag{6.1}$$

With small angle α_1 and α_2, acting force in horizontal direction is assumed to be zero, which means there is no horizontal motion. For the total acting force in the vertical direction, it could be written as:

$$\begin{aligned}\mathbf{F}_v^1 =& T dy \sin\alpha_2 - T dy \sin\alpha_1 \\ \approx& T dy \tan\alpha_2 - T dy \tan\alpha_1\end{aligned} \tag{6.2}$$

Also, we could write tangential term as spatial derivative of wave vector at two different coordinates as:

$$\tan\alpha_1 = \frac{\partial\mathbf{a}(x, y_1, t)}{\partial x} \tag{6.3}$$

$$\tan\alpha_2 = \frac{\partial\mathbf{a}(x+dx, y_2, t)}{\partial x} \tag{6.4}$$

With $y_1, y_2 \in [y, y+dy]$. Similarly, we could perform same analysis towards back and front side of this material and conclude total acting force in vertical direction on these two sides is:

$$\mathbf{F}_v^2 \approx T dx(\frac{\partial\mathbf{a}(x_1, y+dy, t)}{\partial x} - \frac{\partial\mathbf{a}(x_2, y, t)}{\partial x}) \tag{6.5}$$

With $x_1, x_2 \in [x, x+dx]$. In horizontal direction, it is also approximated as zero:

$$\mathbf{F}_h^1 = T dx \cos\alpha_2 - T dx \cos\alpha_1 \approx \mathbf{0} \tag{6.6}$$

From Newton second law, we could formulate dynamic model of wave motion for this small fraction of geological material as:

$$\begin{aligned}m\frac{\partial^2\mathbf{a}(x,y,t)}{\partial t^2} =& \mathbf{F}_h^1 + \mathbf{F}_v^1 + \mathbf{F}_h^2 + \mathbf{F}_v^2 \\ \approx& T dy(\frac{\partial\mathbf{a}(x+dx, y_1, t)}{\partial y} - \frac{\partial\mathbf{a}(x, y_2, t)}{\partial y}) + T dx(\frac{\partial\mathbf{a}(x_1, y+dy, t)}{\partial x} - \frac{\partial\mathbf{a}(x_2, y, t)}{\partial x})\end{aligned} \tag{6.7}$$

Interstingly, we find $m = \rho dx dy$ with ρ is density of material. This finally turns out after neglecting approximation error for very small deflection of material:

$$\begin{aligned}\frac{\partial^2\mathbf{a}(x,y,t)}{\partial t^2} &= \frac{T}{\rho dx}(\frac{\partial\mathbf{a}(x+dx, y_1, t)}{\partial y} - \frac{\partial\mathbf{a}(x, y_2, t)}{\partial y}) + \frac{T}{\rho dy}(\frac{\partial\mathbf{a}(x_1, y+dy, t)}{\partial x} - \frac{\partial\mathbf{a}(x_2, y, t)}{\partial x}) \\ &= \frac{T}{\rho}(\frac{\partial^2\mathbf{a}(x,y,t)}{\partial x^2} + \frac{\partial^2\mathbf{a}(x,y,t)}{\partial y^2})\end{aligned} \tag{6.8}$$

Replace phase velocity of wave c for a P-type seismic wave for poisson solid rock with [13] :

$$\frac{T}{\rho} = \frac{1}{c^2} = \frac{\rho}{\varphi_1 + 2\varphi_2} \tag{6.9}$$

We could finally reach 2D seismic wave equation for homogenous and elastic medium as:

$$\frac{\partial^2\mathbf{a}(x,y,t)}{\partial t^2} = \frac{1}{c^2}(\frac{\partial^2\mathbf{a}(x,y,t)}{\partial x^2} + \frac{\partial^2\mathbf{a}(x,y,t)}{\partial y^2}) \tag{6.10}$$

Receiver distance bound for strict convexity of local cost function
The determinant of Hessian matrix of local cost function is calculated to fulfill strict convexity as:

$$\begin{aligned}det(\mathbf{H}) =&\frac{\partial^2 c_d(\tilde{u}_{d,f},\tilde{t}_{0,d,f})}{\partial \tilde{u}_{d,f}^2}\frac{\partial^2 c_d(\tilde{u}_{d,f},\tilde{t}_{0,d,f})}{\partial \tilde{t}_{0,d,f}^2} - \frac{\partial c_d(\tilde{u}_{d,f},\tilde{t}_{0,d,f})}{\partial \tilde{u}_{d,f}\tilde{t}_{0,d,f}}\frac{\partial c_d(\tilde{u}_{d,f},\tilde{t}_{0,d,f})}{\partial \tilde{t}_{0,d,f}\tilde{u}_{d,f}}\\ =&\frac{1}{2}\frac{t_{d,f}x_d^4}{(\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2)^{\frac{3}{2}}}2(1-\frac{t_{d,f}}{\sqrt{\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2}}+\frac{\tilde{t}_{0,d,f}^2 t_{d,f}}{(\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2)^{\frac{3}{2}}}) - (\frac{t_{d,f}x_d^2\tilde{t}_{0,d,f}}{(\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2)x_d^2)^{\frac{3}{2}}})^2 > 0\end{aligned} \tag{6.11}$$

This turns out:

$$\frac{t_{d,f}x_d^4}{(\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2)^{\frac{3}{2}}}(1-\frac{t_{d,f}}{\sqrt{\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2}}+\frac{\tilde{t}_{0,d,f}^2 t_{d,f}}{(\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2)^{\frac{3}{2}}}) > (\frac{t_{d,f}x_d^2\tilde{t}_{0,d,f}}{(\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2))^{\frac{3}{2}}})^2 \tag{6.12}$$

Divide both sides by $\frac{t_{d,f}x_d^4}{(\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2)^{\frac{3}{2}}}$, we obtain:

$$(1-\frac{t_{d,f}}{\sqrt{\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2}}+\frac{\tilde{t}_{0,d,f}^2 t_{d,f}}{(\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2)^{\frac{3}{2}}}) > (\frac{t_{d,f}\tilde{t}_{0,d,f}^2}{(\tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2))^{\frac{3}{2}}}) \tag{6.13}$$

Rearrange terms and we finally reach:

$$t_{d,f}^2 < \tilde{u}_{d,f}{x_d}^2+\tilde{t}_{0,d,f}^2 \tag{6.14}$$

Which gives us lower bound on source-to-receiver distance:

$$x_d > \sqrt{\frac{t_{d,f}^2-\tilde{t}_{0,d,f}^2}{\tilde{u}_{d,f}}} \tag{6.15}$$

With constraint $t_{d,f} >= \tilde{t}_{0,d,f}$ and $\tilde{u}_{d,f}$ is positive
Derivation of Ricker wavelet spectrum
The Fourier transform of Ricker wavelet is given as:

$$\begin{aligned}\mathcal{F}(r(t)) =&\mathcal{F}((1-\frac{1}{2}\omega_p^2t^2)\exp\left\{-\frac{1}{4}\omega_p^2t^2\right\})\\ =&\mathcal{F}(\exp\left\{-\frac{1}{4}\omega_p^2t^2\right\})+\mathcal{F}(-\frac{1}{2}\omega_p^2t^2\exp\left\{-\frac{1}{4}\omega_p^2t^2\right\})\end{aligned} \tag{6.16}$$

Then we could calculate Fourier transform of first term as:

$$\begin{aligned}\mathcal{F}(\exp\left(-\frac{1}{4}\omega_p^2t^2\right)) &= \int_{-\infty}^{\infty}\exp\left\{-\frac{1}{4}\omega_p^2t^2 - j\omega t\right\}dt\\ &= \int_{-\infty}^{\infty}\exp\left(-\frac{1}{4}\omega_p^2(t^2 - \frac{4j\omega}{\omega_p^2}t)\right)dt\\ &= \int_{-\infty}^{\infty}\exp\left(-\frac{1}{4}\omega_p^2(t - \frac{2j\omega}{\omega_p^2})^2 + \frac{4\omega^2}{\omega_p^4}\right)dt\\ &= \exp\left(-\frac{\omega^2}{\omega_p^2}\right)\int_{-\infty}^{\infty}\exp\left(-\frac{1}{4}\omega_p^2(t - \frac{2j\omega}{\omega_p^2})^2\right)dt\\ &= \frac{2\sqrt{\pi}}{\omega_p}\exp\left(-\frac{\omega^2}{\omega_p^2}\right)\end{aligned} \tag{6.17}$$

For second term, we use following identity:

$$\begin{aligned}\mathcal{F}(-\frac{1}{2}\omega_p^2t^2\exp\left\{-\frac{1}{4}\omega_p^2t^2\right\}) &= \frac{\partial^2\sqrt{\frac{4\pi}{\omega_p^2}}\exp\left(-\frac{\omega^2}{\omega_p^2}\right)}{\partial\omega^2}\\ &= \omega_p\sqrt{\pi}\frac{\partial^2\exp\left(-\frac{\omega^2}{\omega_p^2}\right)}{\partial\omega^2}\\ &= \frac{-2\sqrt{\pi}}{\omega_p}(\exp\left(-\frac{\omega^2}{\omega_p^2}\right) - \exp\left(-\frac{2\omega^2}{\omega_p^2}\right)\frac{\omega^2}{\omega_p^2})\end{aligned} \tag{6.18}$$

Add these two Fourier transform together, we finally get:

$$\mathcal{F}(r(t)) = \frac{4\sqrt{\pi}\omega^2}{\omega_p^3}exp(-\frac{\omega^2}{\omega_p^2}) \tag{6.19}$$

Acronyms

NMO	Normal-moevout
ATC	adapt-then-combine
FDTD	Finite difference time-domain
SGD	Stochastic gradient descent

Bibliography

[1] `https://github.com/scipy/scipy/blob/v1.6.3/scipy/optimize/minpack.py#L532-L834`.

[2] Mussett, A. and Khan, M. A. *"Looking into the earth"*. Chapman & amp Hall.

[3] J. Bérenger. "A Historical Review of the Absorbing Boundary Conditions for Electromagnetics". In *"Forum for Electromagnetic Research Methods and Application Technologies"*.

[4] Yann Le Cun. "EFFICIENT LEARNING AND SECOND-ORDER METHODS". `http://www-labs.iro.umontreal.ca/~vincentp/ift3390/lectures/YannNipsTutorial.pdf`.

[5] Vaezi, Yoones, Van der Baan, Mirko. "Comparison of the STA/LTA and power spectral density methods for microseismic event detection". *Geophysical Journal International*, 203(3):1896–1908, 10 2015.

[6] Jingdong Chen,J.Benesty, Yiteng Huang, S. Doclo. "New insights into the noise reduction Wiener filter". *IEEE Transactions on Audio, Speech and Language Processing*, 14(4):1218–1234, 2006.

[7] Jinyan Fan. "The modified Levenberg-Marquardt method for nonlinear equations with cubic convergence". *Mathematics of Computation*, 81(277):447–466, 2012.

[8] Chen Jianshu, Sayed Ali H. "Diffusion Adaptation Strategies for Distributed Optimization and Learning Over networks". *IEEE Transactions on Signal Processing*, 60(8):4289–4305, 2012.

[9] L. Xiao , S. Boyd ,S. Kim. "Distributed average Consensus with Least-Mean-Square Deviation". In *Journal of Parallel and Distributed Computing*, volume 67, pages 33–46, Juni 2006.

[10] R. Courant, K.Friedrichs,H. Levy. "On the Partial Differential Equations of Mathematical Physics". 1928.

[11] Hans Petter Langtangen. `http://hplgit.github.io/INF5620/doc/pub/main_wave.html`.

[12] Meka Raghu. "cs289ml: Notes on convergence of gradient descent". `https://raghumeka.github.io/CS289ML/gdnotes.pdf`.

[13] Aster Rick. "the seismic wave equation". `http://www.ees.nmt.edu/outside/courses/GEOP523/Docs/waveeq.pdf`, 2011.

[14] Gholamy, Afshin, Kreinovich, Vladik. "Why Ricker Wavelets Are Successful in Processing Seismic data: Towards a theoretical explanation". In *"IEEE Symposium on Computational Intelligence for Engineering Solutions"*, pages 11–16, December 2014.

[15] Yanghua Wang. "Frequencies of the Ricker wavelet". *GEOPHYSICS*, 80(2):31–37, 2015.

[16] E. Staudinger, D. Shutin, C. Manß, A. Viseras, S. Zhang. "Swarm technologies for future space exploration missions". *Proc. I-SAIRAS '18: Int. Symp. Artificial Intelligence, Robotics and Automation in Space*, 2018.