Internship report for the Instituto Colombia del Petróleo (ICP)
In requirement of the degree Master of science Applied Physics, TU Delft, the Netherlands

A general recipe for obtaining adjoint equations and gradients and its application to full waveform inversion of 2D isotropic elastic media in finite difference time-domain

J. J. Wesdorp*

Abstract
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Supervisors
Herling Gonzalez-Alvarez [1]
Prof. Koen van Dongen [2]
Prof. Ana Ramirez [3]

[1] *Instituto Colombia del Petróleo (ICP), Piedecuesta, Colombia*
[2] *Technical University of Delft, the Netherlands*
[3] *Universidad Industrial de Santander (UIS), CPS research group, Bucaramanga, Colombia*
***email**: jaapwesdorp@gmail.com

Contents

1 Introduction **2**

2 A simple recipe for the adjoint state method **3**

2.1 Adjoint method for general discretized linear systems . . . 3
Formulation • Gradients and adjoint • Least squares cost function

2.2 Adjoint and gradients of the elastic wave equation . . . 4

3 Numerical implementation of the elastic wave equation **6**

3.1 Staggered grid discrete equations . . . 6

3.2 Ricker wavelet source function . . . 7

3.3 CPML Absorbing boundaries . . . 9
Discussion on free CPML parameters

3.4 Obtaining realistic model parameters . . . 11
Brocher relations

3.5 Total system energy . . . 11

3.6 Free surface boundary conditions . . . 11

4 Implementation Full Waveform Inversion **12**

4.1 Adjoint back-propagation . . . 12

4.2 Gradient calculation . . . 13

4.3 Model updating using L-BFGS gradient descent . . . 14

4.4 Implementation in C . . . 15
Memory considerations

5 FWI results **17**

5.1 Rescaled version of realistic seismic testing model . . . 23

6 Conclusion **25**

Appendices **25**

A Lagrangian based derivation of the gradient via the adjoint state method **25**

A.1 General system of equations . . . 25

A.2 Derivative to model parameters . . . 25

B More example applications **27**

B.1 The second order electromagnetic wave equation . . . 27

B.2 The second order acoustic wave equation . . . 28

B.3 The first order acoustic wave equation . . . 28

C Rayleigh wave free surface test **30**

References **31**

1. Introduction

Full waveform inversion can be used a tool to quantitatively image material properties of the earth, or tissue.

30 years ago -> [1] proposed FWI. Now since 10 years the industry is making use of FWI in their processing workflow due to the gain in computing power.

Initially mostly acoustic, but there is interest in modeling elastic waves, since you can see phenomena like: groundroll waves, surface waves ->(WHAT ELSE IS SPECIAL ABOUT ELASTIC).

FWI requires one to know the gradient of your guessed model with respect to measured shots, therefore the *adjoint state method* [2] makes this a lot more efficient.

This research was performed as part of an internship for the Instituto Colombia del Petróleo (ICP). The first goal was to model the isotropic 2D elastic wave equation, which is described in section ... The second goal was to find the adjoint operator and gradients for the elastic case. Since no explicit declaration is given yet for the elastic case in current literature and the adjoint state method is usually described in a very abstract way, this work shows an "engineers approach" to obtaining the adjoint system and gradient expressions for any general set of equations in sec .... This method is applied to the elastic case in sec ..., and more examples are given in Appendix B. This method is subsequently tested by performing FWI on a test model described in sec ...

2. A simple recipe for the adjoint state method

2.1 Adjoint method for general discretized linear systems

This section shows in simple steps how to obtain the adjoint equations and gradients given a general system of wave equations and initial and boundary conditions of zero. These are cast in a general form usually present in acoustic, elastic or electromagnetic FWI applications. The proof of validity and derivation of this method is presented in Appendix A for an more general set of equations based on the notes of Bradley[3].

2.1.1 Formulation

Say we have a model representing the earth (in seismic applications) or tissue (in medical applications) for some parameter like density ρ or speed of sound v, which we discretized in a grid in space into $N_g (= N_x \times N_z$ in 2D) points. If we want to model d_m parameters over this grid we can write the values of these parameters over the grid as a vector $\vec{m}$ of dimension $d_m \times N_g$. We want to model d_s fields also discretized in space on the same grid we can write them as a vector $\vec{s}$ of dimension $d_s \times N_g$. Any FWI problem of wave equations(e.g electromagnetic, elastic or acoustic), whether they are multiple first or second order PDE's in time and space, can be written as the following minimization problem given a set of discretized(only in space) ODEs

$$\begin{aligned} \min_{\vec{m}} \ \chi(\vec{s},\vec{m}) &= \int_0^T f(\vec{s},\vec{m})\,dt \\ \text{subject to} \quad & T(\vec{m})\ddot{\vec{s}} - C(\vec{m})\dot{\vec{s}} - A(\vec{m})\vec{s} - \vec{b}(\vec{m}) = 0 \\ \text{with B.C} \quad & \vec{s}(0) = 0 \\ & \dot{\vec{s}}(0) = 0 \end{aligned} \tag{1}$$

where T, C, A are big matrices of $(N_g \times d_s) \times (N_g \times d_s)$ coefficients which follow from the discretization in space of a PDE system, $\vec{b}(\vec{m})$ is a vector of the same length of $\vec{s}$ which contains the source terms, χ denotes the total cost function and we denote time explicitely by taking the integral of some function f since usually we simplify each cost function into this form.

2.1.2 Gradients and adjoint

The goal of the adjoint state method is avoiding the calculation of the Frechet derivatives $\frac{\partial}{\partial \chi} m$ since they require the amount of forward modeling evaluations to scale with the grid size of the model. We can apply Eq. (40) of Appendix A to the system of Eq. (1) and find the following simpler expression for the total gradient with respect to *all* model parameters

$$\frac{d\chi}{d\vec{m}} = \int_0^T \frac{\partial f}{\partial \vec{m}} + \vec{\lambda}^T \left(\frac{\partial T}{\partial \vec{m}} \ddot{\vec{s}} - \frac{\partial C}{\partial \vec{m}} \dot{\vec{s}} - \frac{\partial A}{\partial \vec{m}} \vec{s} - \frac{\partial b}{\partial \vec{m}} \right) dt \tag{2}$$

Note that the gradients $\frac{\partial T}{\partial \vec{m}}, \frac{\partial C}{\partial \vec{m}}, \frac{\partial A}{\partial \vec{m}}$ can be seen as a sum of all the $3N$ derivatives of each component of the matrices $T(\vec{m}), C(\vec{m}), A(\vec{m})$ to the discrete variables $\vec{m}_i$, but in practice we can write this into simple compact expressions as shown in the examples in Appendix B. The adjoint variable λ is given using Eq. (39) and applying it to our system of Eq. (1) we obtain

$$T^T(\vec{m}) \ddot{\vec{\lambda}} = -C^T(\vec{m}) \dot{\vec{\lambda}} + A^T(\vec{m}) \vec{\lambda} - \frac{\partial f}{\partial \vec{s}} \tag{3}$$

giving an extra set of equations that need to be solved.

2.1.3 Least squares cost function

In most cases the cost function is given by the misfit between observed data d at certain receiver positions y_R and the modeled data at the same positions s_{y_R}, so $f(\vec{s})$ is given by

$$f(\vec{s}) = \frac{1}{2} \sum_R^{N_R} ||s_{y_R} - d_{y_R}||^2 \tag{4}$$

where the sum goes over all receiver positions. Note that $f(\vec{s})$ does not depend directly on the model parameters $\vec{m}$ making the gradient expression simpler since $\frac{\partial f}{\partial \vec{m}} = 0$. We thus obtain

$$\frac{\partial f}{\partial \vec{s}} = \sum_R^{N_R} s_{y_R} - d_{y_R} \tag{5}$$

which are the well known residuals injected as a source for the adjoint equations.

2.2 Adjoint and gradients of the elastic wave equation

Elastic waves in 2D isotropic media can be modeled by using only three parameters, the density ρ and the two Lame parameters μ, λ and follow the following system of equations

$$\begin{aligned}
\rho \frac{\partial v_x}{\partial t} &= \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xz}}{\partial z} \\
\rho \frac{\partial v_z}{\partial t} &= \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{zz}}{\partial z} \\
\frac{\partial \tau_{xx}}{\partial t} &= (\lambda + 2\mu) \frac{\partial v_x}{\partial x} + \lambda \frac{\partial v_z}{\partial z} + b_{xx} \\
\frac{\partial \tau_{zz}}{\partial t} &= (\lambda + 2\mu) \frac{\partial v_z}{\partial z} + \lambda \frac{\partial v_x}{\partial x} + b_{zz} \\
\frac{\partial \tau_{xz}}{\partial t} &= \mu \left(\frac{\partial v_x}{\partial z} + \frac{\partial v_z}{\partial x} \right)
\end{aligned} \tag{6}$$

where we only added the source to the stress as to resemble an earthquake. The form of b is independent of the model parameters and usually taken as a ricker wavelet. writing this in the form of Eq. (1) gives

$$T(m)=0,\ s=\begin{bmatrix} v_x \\ v_z \\ \tau_{xx} \\ \tau_{zz} \\ \tau_{xz} \end{bmatrix},\ C(m)=-\begin{bmatrix} \rho & 0 & 0 & 0 & 0 \\ 0 & \rho & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{7}$$

and

$$A(m)=\begin{bmatrix} 0 & 0 & \mathscr{D}_x & 0 & \mathscr{D}_z \\ 0 & 0 & 0 & \mathscr{D}_z & \mathscr{D}_x \\ (\lambda+2\mu)\mathscr{D}_x & \lambda\mathscr{D}_z & 0 & 0 & 0 \\ \lambda\mathscr{D}_x & (\lambda+2\mu)\mathscr{D}_z & 0 & 0 & 0 \\ \mu\mathscr{D}_z & \mu\mathscr{D}_x & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

Thus we have $C^T(m)=C(m)$ and

$$A^T(m)=\begin{bmatrix} 0 & 0 & -\mathscr{D}_x(\lambda+2\mu) & -\mathscr{D}_x\lambda & -\mathscr{D}_z\mu \\ 0 & 0 & -\mathscr{D}_z\lambda & -\mathscr{D}_z(\lambda+2\mu) & -\mathscr{D}_x\mu \\ -\mathscr{D}_x & 0 & 0 & 0 & 0 \\ 0 & -\mathscr{D}_z & 0 & 0 & 0 \\ -\mathscr{D}_z & -\mathscr{D}_x & 0 & 0 & 0 \end{bmatrix} \tag{9}$$

which in turn (when transforming back to the continuous domain from the discrete variables) results in the following adjoint equations(naming the adjoint variable l) as defined in Eq. (3)

$$\begin{aligned} \rho\frac{\partial l_1}{\partial t} &= \frac{\partial((\lambda+2\mu)l_3)}{\partial x}+\frac{\partial(\lambda l_4)}{\partial x}+\frac{\partial(\mu\lambda_5)}{\partial z} \\ \rho\frac{\partial l_2}{\partial t} &= \frac{\partial(\lambda l_3)}{\partial z}+\frac{\partial((\lambda+2\mu)l_4)}{\partial z}+\frac{\partial(\mu\lambda_5)}{\partial x} \\ \frac{\partial l_3}{\partial t} &= \frac{\partial l_1}{\partial x} \\ \frac{\partial l_4}{\partial t} &= \frac{\partial l_2}{\partial z} \\ \frac{\partial l_5}{\partial t} &= \frac{\partial l_1}{\partial z}+\frac{\partial l_2}{\partial x} \end{aligned} \tag{10}$$

where the order of differentiation and multiplication with the model parameters is important. The gradients then follow from Eq. (2), giving

$$
\begin{aligned}
\frac{d\chi}{d\rho} &= \int_0^T -l^T \frac{\partial C}{\partial \rho} \dot{s}\, dt \\
&= \int_0^T \begin{bmatrix} \vec{l_1} & \vec{l_2} & \vec{l_3} & \vec{l_4} & \vec{l_5} \end{bmatrix} \begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \dot{\vec{v}_x} \\ \dot{\vec{v}_y} \\ \dot{\vec{\tau}_{xx}} \\ \dot{\vec{\tau}_{zz}} \\ \dot{\vec{\tau}_{xz}} \end{bmatrix} \\
&= \int_0^T \vec{l_1}\dot{\vec{v}_x} + \vec{l_2}\dot{\vec{v}_z}\, dt
\end{aligned} \tag{11}
$$

,

$$
\begin{aligned}
\frac{d\chi}{d\lambda} &= \int_0^T -l^T \frac{\partial A}{\partial \lambda} s\, dt \\
&= -\int_0^T \begin{bmatrix} \vec{l_1} & \vec{l_2} & \vec{l_3} & \vec{l_4} & \vec{l_5} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \mathscr{D}_x & \mathscr{D}_z & 0 & 0 & 0 \\ \mathscr{D}_x & \mathscr{D}_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{v_x} \\ \vec{v_y} \\ \vec{\tau_{xx}} \\ \vec{\tau_{zz}} \\ \vec{\tau_{xz}} \end{bmatrix} \\
&= -\int_0^T \left(\vec{l_3} + \vec{l_4}\right) \left(\frac{\partial \vec{v_x}}{\partial x} + \frac{\partial \vec{v_z}}{\partial z}\right)
\end{aligned} \tag{12}
$$

and

$$
\begin{aligned}
\frac{d\chi}{d\mu} &= \int_0^T -l^T \frac{\partial A}{\partial \mu} s\, dt \\
&= -\int_0^T \begin{bmatrix} \vec{l_1} & \vec{l_2} & \vec{l_3} & \vec{l_4} & \vec{l_5} \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 2\mathscr{D}_x & 0 & 0 & 0 & 0 \\ 0 & 2\mathscr{D}_z & 0 & 0 & 0 \\ \mathscr{D}_z & \mathscr{D}_x & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \vec{v_x} \\ \vec{v_y} \\ \vec{\tau_{xx}} \\ \vec{\tau_{zz}} \\ \vec{\tau_{xz}} \end{bmatrix} \\
&= -\int_0^T 2\vec{l_3}\frac{\partial \vec{v_x}}{\partial x} + 2\vec{l_4}\frac{\partial \vec{v_z}}{\partial z} + \vec{l_5}\left(\frac{\partial \vec{v_x}}{\partial z} + \frac{\partial \vec{v_z}}{\partial x}\right) dt
\end{aligned} \tag{13}
$$

3. Numerical implementation of the elastic wave equation

Chapter intro here

3.1 Staggered grid discrete equations

We now show the modeling of Eq. (6). Due to the nature of the equations, if we would implement every field at the same grid point in time and space using centered finite differences, we would get decoupling between the velocity and stress in time and space(e.g V_x at time t depending on τ at time $t \pm \frac{dt}{2}$ which we do not have defined). This could be solved by using a two times finer grid, but can be more elegantly solved by use a staggered grid following [4]. Using the notation $\{v_x, v_z, \tau_{xx}, \tau_{zz}, \tau_{xz}\} = \{U, V, X, Z, T\}$ we do a "leapfrog" scheme, which in each timestep loops over

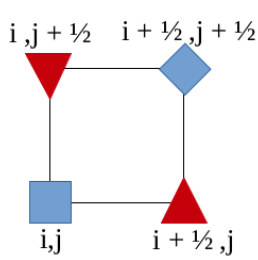
the spatial directions for calculating $U^{n+\frac{1}{2}}, V^{n+\frac{1}{2}}$ as a function of the previously calculated X^n, Z^n, T^n first and consecutively loops over the spatial directions again to calculate $X^{n+1}, Z^{n+1}, T^{n+1}$ depending on the just calculated $U^{n+\frac{1}{2}}, V^{n+\frac{1}{2}}$. This avoids the decoupling mentioned above, but requires the fields to be defined in different points in the grid. The numerical equations are then given by (using a 2nd order central difference in time and arbitrary difference operator in space)

$$\begin{cases} U_{i,j}^{n+\frac{1}{2}} = U_{i,j}^{n-\frac{1}{2}} + \dfrac{\Delta t}{\rho_{i,j}} \left[D_x X_{i,j}^n + D_z T_{i,j}^n \right] \\ V_{i+\frac{1}{2},j+\frac{1}{2}}^{n+\frac{1}{2}} = V_{i+\frac{1}{2},j+\frac{1}{2}}^{n-\frac{1}{2}} + \dfrac{\Delta t}{\rho_{i+\frac{1}{2},j+\frac{1}{2}}} \left[D_x T_{i+\frac{1}{2},j+\frac{1}{2}}^n + D_z Z_{i+\frac{1}{2},j+\frac{1}{2}}^n \right] \\ X_{i+\frac{1}{2},j}^{n+1} = X_{i+\frac{1}{2},j}^n + \Delta t \left[\left(\lambda_{i+\frac{1}{2},j} + 2\mu_{i+\frac{1}{2},j} \right) D_x U_{i+\frac{1}{2},j}^{n+\frac{1}{2}} + \lambda_{i+\frac{1}{2},j} D_z V_{i+\frac{1}{2},j}^{n+\frac{1}{2}} \right] \\ Z_{i+\frac{1}{2},j}^{n+1} = Z_{i+\frac{1}{2},j}^n + \Delta t \left[\left(\lambda_{i+\frac{1}{2},j} + 2\mu_{i+\frac{1}{2},j} \right) D_z V_{i+\frac{1}{2},j}^{n+\frac{1}{2}} + \lambda_{i+\frac{1}{2},j} D_x U_{i+\frac{1}{2},j}^{n+\frac{1}{2}} \right] \\ T_{i,j+\frac{1}{2}}^{n+1} = T_{i,j+\frac{1}{2}}^n + \mu_{i,j+\frac{1}{2}} \Delta t \left[D_z U_{i,j+\frac{1}{2}}^{n+\frac{1}{2}} + D_x V_{i,j+\frac{1}{2}}^{n+\frac{1}{2}} \right] \end{cases} \tag{14}$$

where for the simplest second order differential operator $D_x A_{i,j}^n = \frac{A_{i+\frac{1}{2},j}^n - A_{i-\frac{1}{2},j}^n}{\Delta_x}$ and $D_z A_{i,j}^n = \frac{A_{i,j+\frac{1}{2}}^n - A_{i,j-\frac{1}{2}}^n}{\Delta_z}$. n denotes the discrete time step, i, j the x, z coordinates respectively, and $\Delta t, \Delta x, \Delta z$ are the stepsizes taken into each direction respectively. In Fig. 1 the staggered grid is shown schematically together with a table to explain the index translations used to transform from the actual implementation to the discrete equations shown above.

Figure 1. (a) Schematic of the staggered grid. The square indicates the position of the velocity field U, the diamond that of V, the triangle that of X, Z and the upside down triangle that of T. Blue indicates that the field is evaluated at time $n+1/2$ and red at time n. (b) Table of the staggered grid computational array index conversion. Shows how the 2D computational array indexes for the field arrays are transformed for the staggered grid. For example: if we access array element $V[i, j]$ in the code we obtain the field v_z at real position $i+1/2, j+1/2$ in the grid.

(a)



(b)

Field	index i transform	index j transform
$U(v_x)$	i	j
$V(v_z)$	$i+1/2$	$j+1/2$
$X(\tau_{xx})$	$i+1/2$	j
$Z(\tau_{zz})$	$i+1/2$	j
$T(\tau_{xz})$	i	$j+1/2$

3.2 Ricker wavelet source function

We used a source of the form

$$f(t) = \left(1 - 2\pi^2 f_q^2 (t-t_0)^2\right) e^{-\pi^2 f_q^2 (t-t_0)^2} \tag{15}$$

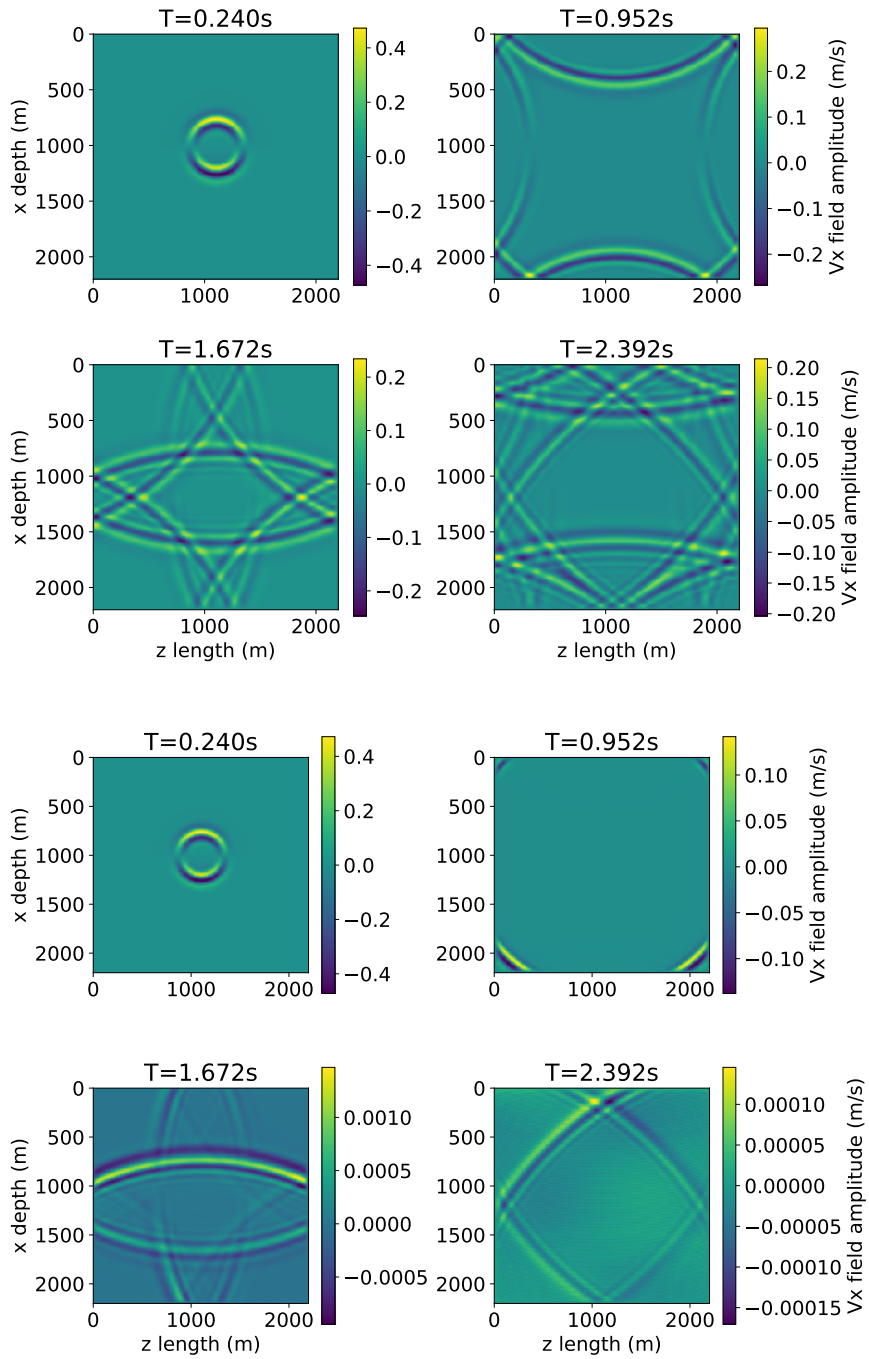


Figure 2. Elastic wave propagation in homogeneous media. Top 4 images show the simulation with reflecting boundary conditions and the bottom four with CPML active, giving a clear decrease in reflection amplitude. $V_p = 2500$m/s, $V_s = 430$m/s, $\rho = 1750$m/s, $f_{source} = 10$Hz, $n_x = n_z = 220$, $d_x = d_z = 10$m, $dt = 0.3$ms

where f_q is the desired center source frequency, t_0 the time offset (taken as $\frac{1}{f_q}$ usually). Note that t_0 is important since if you start with a non-zero source the FD modeling will become unstable. The source was added equally to both the horizontal and vertical stress τ_{xx} and τ_{zz} at each timestep.

3.3 CPML Absorbing boundaries

In most seismic cases we have an infinite domain of our model, but we are just interested in a specific part of this domain. In order to simulate only this part without reflecting boundaries one needs to simulate "open" boundaries. This is performed using Perfectly Matched Layers(PML) [5]. Simply stated PML can be seen as adding as changing the coordinates to complex values in the boundary layers of the grid in which the wave travels, this then exponentially dampens any plane wave traveling through the boundary layer of which the strength depends on the incidence angle, damping less for grazing incidence. Another property of PML is that the reflection coefficient is exactly zero for all incident angles before discretization. The original PML required to extra fields in time, so therefore the Convolutionary PML was introduced(CPML) [6], which allows for the use of memory variables to keep a memory efficient solution and finally and improved version of CPML was introduced to decrease the reflections at grazing incidence angles at the cost of adding extra tunable parameters [7].

CPML can be easily implemented, since all one needs to do is replace all the finite difference operations on the fields inside the CPML layer by CPML damped variant. This is performed by adding so called memory variables which have their own simple governing differential equations. For the field $K \in \{U,V,X,Z,T\}$ and derivative direction $s \in \{x,z\}$ this becomes

$$D_s K^n_{i,j} \rightarrow \frac{D_s K^n_{i,j}}{\kappa_s} + M^n_{D_s K|i,j} \tag{16}$$

where κ_s is usually taken as 1 and the memory variable is updated each timestep by

$$M^n_{D_s K|i,j} = b_s M^{n-1}_{D_s K|i,j} + a_s D_s K^n_{i,j} \tag{17}$$

the damping coefficients a_s, b_s are given by the following formulas

$$\begin{cases} a_s = \dfrac{d_s}{d_s + \alpha_s}(b_s - 1) \\ b_s = e^{-(d_s+\alpha_s)\Delta t} \end{cases} \tag{18}$$

where all parameters are chosen following [7]. They pick α_s a linearly varying from $\alpha_{max} = \pi f_0$ in the entry of the CPML layer to 0 at the end of the layer and f_0 is the center source frequency. $d_s = d_0 \left(\frac{\Delta_{cpml} s}{L}\right)^N$ where $\Delta_{cpml} s$ Is the amount of meters inside the CPML layer in direction s, L is the total length of the CPML layer (usually taken as the length of 10 grid points), N is the damping order (taken as 2) [8], d_0 is given by $d_0 = -(N+1)\, v_p \frac{\log(R_c)}{2L} \approx 341.9$ when we take a theoretical reflection coefficient $R_c = 0.001$ [8]. Note that attention needs to be payed to the values of a_s and b_s at half grid points of the staggered grid, there they should be interpolated as stressed in [7]. For completeness

the full discrete equations are given below inside the CPML layer

$$
\begin{cases}
U_{i,j}^{n+\frac{1}{2}} = U_{i,j}^{n-\frac{1}{2}} + \dfrac{\Delta t}{\rho_{i,j}} \left[\dfrac{D_x X_{i,j}^n}{\kappa_{x_i}} + M_{D_x X|i,j}^{n+\frac{1}{2}} + \dfrac{D_z T_{i,j}^n}{\kappa_{z_j}} + M_{D_z T|i,j}^{n+\frac{1}{2}} \right] \\
V_{i+\frac{1}{2},j+\frac{1}{2}}^{n+\frac{1}{2}} = V_{i+\frac{1}{2},j+\frac{1}{2}}^{n-\frac{1}{2}} + \dfrac{\Delta t}{\rho_{i+\frac{1}{2},j+\frac{1}{2}}} \left[\dfrac{D_x T_{i+\frac{1}{2},j+\frac{1}{2}}^n}{\kappa_{x_{i+\frac{1}{2}}}} + M_{D_x T|i+\frac{1}{2},j+\frac{1}{2}}^{n+\frac{1}{2}} + \dfrac{D_z Z_{i+\frac{1}{2},j+\frac{1}{2}}^n}{\kappa_{z_{j+\frac{1}{2}}}} + M_{D_z Z|i+\frac{1}{2},j+\frac{1}{2}}^{n+\frac{1}{2}} \right] \\
X_{i+\frac{1}{2},j}^{n+1} = X_{i+\frac{1}{2},j}^n + \Delta t \left[\left(\lambda_{i+\frac{1}{2},j} + 2\mu_{i+\frac{1}{2},j} \right) \left(\dfrac{D_x U_{i+\frac{1}{2},j}^{n+\frac{1}{2}}}{\kappa_{x_{i+\frac{1}{2}}}} + M_{D_x U|i+\frac{1}{2},j}^{n+1} \right) + \lambda_{i+\frac{1}{2},j} \left(\dfrac{D_z V_{i+\frac{1}{2},j}^{n+\frac{1}{2}}}{\kappa_{z_j}} + M_{D_z V|i+\frac{1}{2},j}^{n+1} \right) \right] \\
Z_{i+\frac{1}{2},j}^{n+1} = Z_{i+\frac{1}{2},j}^n + \Delta t \left[\left(\lambda_{i+\frac{1}{2},j} + 2\mu_{i+\frac{1}{2},j} \right) \left(\dfrac{D_z V_{i+\frac{1}{2},j}^{n+\frac{1}{2}}}{\kappa_{z_j}} + M_{D_z V|i+\frac{1}{2},j}^{n+1} \right) + \lambda_{i+\frac{1}{2},j} \left(\dfrac{D_x U_{i+\frac{1}{2},j}^{n+\frac{1}{2}}}{\kappa_{x_{i+\frac{1}{2}}}} + M_{D_x U|i+\frac{1}{2},j}^{n+1} \right) \right] \\
T_{i,j+\frac{1}{2}}^{n+1} = T_{i,j+\frac{1}{2}}^n + \mu_{i,j+\frac{1}{2}} \Delta t \left[\dfrac{D_z U_{i,j+\frac{1}{2}}^{n+\frac{1}{2}}}{\kappa_{z_{j+\frac{1}{2}}}} + M_{D_z U|i,j+\frac{1}{2}}^{n+1} + \dfrac{D_x V_{i,j+\frac{1}{2}}^{n+\frac{1}{2}}}{\kappa_{x_i}} + M_{D_x V|i,j+\frac{1}{2}}^{n+1} \right]
\end{cases}
\tag{19}
$$

3.3.1 Discussion on free CPML parameters

The amount of extra tunable parameters CPML seems to give looks overwhelming, but after following the recipe the only tunable parameters that remain are R_c, L, κ and α_{max},κ is taken as a constant $\kappa = 1$ (although [6] introduced a variable κ because it can have an effect on evanescent waves in the electromagnetic case, in the seismic case [7] state that is of little influence and therefore taken constant) and $\alpha_{max} = \pi f_0$ depends on the source so is not really free at the end. This results solely in having to choose R_c and L which in this research always remained at $R_c = 0.001$ and $L = 10\Delta_x$ eliminating the need for retuning the CPML variables during FWI. Although it's good practice to keep checking if the system energy really drops by at least a few orders of magnitude when turning on CPML as shown in Fig. 3.

Algorithm 1 Forward modeling

```
function PROPAGATE FORWARD( )
    X ← SOURCE(0),  Z ← SOURCE(0)          ▷ add initial earthquake stress source for τxx, τzz
    for it = 1 and it < Nt do
        U[it], V[it] ← PROPAGATE VELOCITY(X, Z, T)                          ▷ Step Eq. (19)
        X, Z, T ← PROPAGATE STRESS(U[it], V[it])
        X ← SOURCE(it),  Z ← SOURCE(it)        ▷ add earthquake stress source for τxx, τzz
        it ← it + 1
    end for
end function
```

3.4 Obtaining realistic model parameters

Modeling elastic waves requires knowledge of all three model parameters: the density ρ, and lamé parameters μ and λ. Usually in literature, instead of the Lamé parameters, the S and P-wave velocities are given since these can be read more inituitively. There are simple relations between the wavespeeds and Lamé parameters following from the elastic wave equations

$$\{\rho,\mu,\lambda\} \rightarrow \{\rho,v_s,v_p\}, \text{ with } v_s = \sqrt{\frac{\mu}{\rho}},\ v_p = \sqrt{\frac{\lambda+2\mu}{\rho}} \tag{20}$$

3.4.1 Brocher relations

Usually seismic tomography and refraction studies of the earth crust report only the P-wave velocity v_p, while we need the density and S-wave velocity as well. In order to obtain realistic estimates of these parameters we can the relations derived by T. M. Brocher in 2005 [9] relating v_p to v_s and v_p to ρ. These are derived by studying many real world datasets of the earth's crust giving an empirical method to obtain ρ and v_s given known v_p(in km/s)

$$\rho[g/cm^3] = 1.6612v_p - 0.4721v_p^2 + 0.0671v_p^3 - 0.0043v_p^4 + 0.000106v_p^5 \tag{21}$$

and

$$v_s[km/s] = 0.7857 - 1.2344v_p + 0.7949v_p^2 - 0.1238v_p^3 + 0.0064v_p^4 \tag{22}$$

which are only valid between 1.5 km/s $< v_p < 8$ km/s. This then allows us to create synthetic models of all other parameters given that we have a model for v_p.

3.5 Total system energy

In any wave equation modeling, an important check is to measure the system energy, and see that if there is no source active and no absorbing boundaries, energy remains constant in the system. For elastic waves the energy is given by a combination of kinetic energy K of the moving particles and spring energy T stored by displacing these particles from their equilibrium position following from Hooke's law.

$$E = K + T = \frac{1}{2}\rho||v||^2 + \frac{1}{2}\sum \tau_{ij}\varepsilon_{ij} \tag{23}$$

3.6 Free surface boundary conditions

Until now the boundary conditions was either assumed infinite or reflecting. In reality one often encounters free surface boundary conditions which is defined by

$$\begin{cases} \tau_{xx} = 0 \\ \tau_{xz} = 0 \end{cases} \tag{24}$$

at the boundary and signifies that there is no normal and shear stress at the surface. The waves resulting from this boundary conditions are called Rayleigh-waves after their discoverer which travel along the surface with a speed much slower than the P and S waves. The boundary conditions in this work were simply implemented by setting the stresses at the boundary explicitly to zero. This works for planar surface conditions but gives instabilities when more complex surface topologies are considered. A solution to this would be to use either a stress-image technique [10], or in the case of a interface with vacuum one can use a more advanced parameter averaging scheme [11].

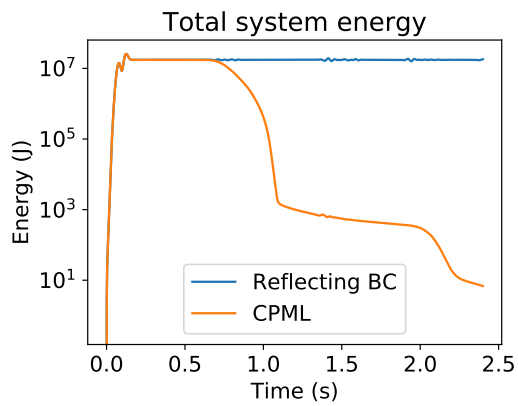


Figure 3. Energy versus time with and without CPML boundaries for the plots of Fig. 2. CPML parameters used are $R = 0.001$, $\kappa = 1$, $n_{cpml} = 10$. The first drop of energy is when the wave reaches the boundary, the second drop is due to the residual reflections reaching the boundary once more.

4. Implementation Full Waveform Inversion

4.1 Adjoint back-propagation

To avoid approximating $\frac{\partial \chi}{\partial dm}$ numerically by evaluating the forward model at least $2 \times N_x \times N_z$ times we use the adjoint state method as described in Section 2.1. The discrete version of our derived set of continuous equations given in Eq. (10) is for a central 2nd order in time and arbitrary order in space by

$$
\begin{aligned}
\overline{U}^{n-\frac{1}{2}}_{i,j} &= \overline{U}^{n+\frac{1}{2}}_{i,j} - \frac{\Delta t}{\rho_{i,j}} \left(D_x \left[(\lambda_{i,j} + 2\mu_{i,j}) \overline{X}^n_{i,j} \right] + D_x \left[\lambda_{i,j} \overline{Z}^n_{i,j} \right] + D_z \left[\mu_{i,j} \overline{T}^n_{i,j} \right] \right) \\
\overline{Z}^{n-\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2}} &= \overline{Z}^{n+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2}} - \frac{\Delta t}{\rho_{i+\frac{1}{2},j+\frac{1}{2}}} \left(D_z \left[\lambda_{i+\frac{1}{2},j+\frac{1}{2}} \overline{X}^n_{i+\frac{1}{2},j+\frac{1}{2}} \right] + D_z \left[\left(\lambda_{i+\frac{1}{2},j+\frac{1}{2}} + 2\mu_{i+\frac{1}{2},j+\frac{1}{2}} \right) \overline{Z}^n_{i+\frac{1}{2},j+\frac{1}{2}} \right] \right. \\
&\qquad \left. + D_x \left[\mu_{i+\frac{1}{2},j+\frac{1}{2}} \overline{T}^n_{i+\frac{1}{2},j+\frac{1}{2}} \right] \right) \\
\overline{X}^n_{i+\frac{1}{2},j} &= \overline{X}^{n+1}_{i+\frac{1}{2},j} - \Delta t D_x \left[\overline{U}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} \right] \\
\overline{Z}^n_{i+\frac{1}{2},j} &= \overline{Z}^{n+1}_{i+\frac{1}{2},j} - \Delta t D_z \left[\overline{V}^{n+\frac{1}{2}}_{i+\frac{1}{2},j} \right] \\
\overline{T}^n_{i,j+\frac{1}{2}} &= \overline{T}^{n+1}_{i,j+\frac{1}{2}} - \Delta t \left(D_z \left[\overline{U}^{n+\frac{1}{2}}_{i,j+\frac{1}{2}} \right] + D_x \left[\overline{V}^{n+\frac{1}{2}}_{i,j+\frac{1}{2}} \right] \right)
\end{aligned}
\tag{25}
$$

where the adjoint versions of the corresponding forward fields are denoted with a line above the letter. The discrete differential operators in direction $s \in \{x, z\}$ are denoted by $D_s[\circ]$. Note that it is important that the parameters are multiplied with the fields *before* the differentiation is applied. For example with a 2nd order central finite difference $D_z \left[\lambda_{i+\frac{1}{2},j+\frac{1}{2}} \overline{X}^n_{i+\frac{1}{2},j+\frac{1}{2}} \right] = \frac{1}{\Delta z} \left(\lambda_{i+\frac{1}{2},j+1} \overline{X}^n_{i+\frac{1}{2},j+1} - \lambda_{i+\frac{1}{2},j} \overline{X}^n_{i+\frac{1}{2},j} \right)$. Also note the swapping and sign difference of $n, n+1$ with respect to the forward equations since we

are backpropagating from $t = T_{final}$ to 0.

A CPML layer was also implemented for the adjoint equations. The method to derive the equations inside the CPML layer was the same as for the forward equations and simply consisted of adding memory variables to each derivative present in Eq. (25). The parameters for CPML used where the same as in the forward model and this thus resulted in a total of 8 extra 2D fields (memory variables) being stored in the adjoint CPML area.

Algorithm 2 Backward adjoint modeling and gradient calculation

function PROPAGATE ADJOINT AND CALCULATE GRADIENTS(U, V) ▷ Beginning of adjoint loop
  $\gamma_\rho = 0, \gamma_\mu = 0, \gamma_\lambda = 0$ ▷ Initialize gradients to 0
  $\overline{U} \leftarrow \overline{U} - \left(d_U^{mod}(N_t - 1) - d_U^{meas}(N_t - 1)\right)$ ▷ Eq. (5)
  $\overline{Z} \leftarrow \overline{Z} - \left(d_Z^{mod}(N_t - 1) - d_Z^{meas}(N_t - 1)\right)$
  for $it = 1$ to $it < N_t$ **do**
    $\overline{X}, \overline{Z}, \overline{T} \leftarrow$ PROPAGATE ADJOINT STRESS($\overline{U}, \overline{V}$) ▷ Step Eq. (25)
    $\overline{U}, \overline{V} \leftarrow$ PROPAGATE ADJOINT VELOCITY($\overline{X}, \overline{Z}, \overline{T}$)
    ▷ add residuals from Eq. (5) as source to the adjoint fields of V_x and V_z
    $\overline{U} \leftarrow \overline{U} - \left(d_U^{mod}(N_t - 1 - it) - d_U^{meas}(N_t - 1 - it)\right)$ ▷ Eq. (5)
    $\overline{Z} \leftarrow \overline{Z} - \left(d_Z^{mod}(N_t - 1 - it) - d_Z^{meas}(N_t - 1 - it)\right)$
    $it \leftarrow it + 1$
    $\gamma_\rho, \gamma_\mu, \gamma_\lambda \leftarrow$STEP GRADIENTS($it, U, V, \overline{U}, \overline{V}, \overline{X}, \overline{Z}, \overline{T}$) ▷ Add steps of the sum of eq Eqs. (26) to (28) to the gradients
  end for
end function

4.2 Gradient calculation

The gradient is calculated following Eq. (2) using the same notation as in Eqs. (19) and (25). A 2nd order central difference in space and time was used. For ρ we get

$$\frac{\partial \chi}{\partial \rho} \approx \sum_{n=0}^{N_t-1} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_z-1} \frac{1}{2} \left[\overline{U}_{i,j}^{n+\frac{1}{2}} \left(U_{i,j}^{n+1\frac{1}{2}} - U_{i,j}^{n-\frac{1}{2}} \right) + \overline{V}_{i+\frac{1}{2},j+\frac{1}{2}}^{n+\frac{1}{2}} \left(V_{i+\frac{1}{2},j+\frac{1}{2}}^{n+1\frac{1}{2}} - V_{i+\frac{1}{2},j+\frac{1}{2}}^{n-\frac{1}{2}} \right) \right] \Delta x \Delta z \tag{26}$$

where we used a 2nd order in time central difference of $2\Delta t$ to make the fields coincide with our choice of staggered grid. For λ we get

$$\begin{aligned} \frac{\partial \chi}{\partial \lambda} \approx \sum_{n=0}^{N_t-1} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_z-1} \left(\overline{X}_{i+\frac{1}{2},j}^{n} + \overline{Z}_{i+\frac{1}{2},j}^{n} \right) \Bigg[\frac{1}{2\Delta x} \Bigg(\left(U_{i+1,j}^{n-\frac{1}{2}} + U_{i+1,j}^{n+\frac{1}{2}} \right) - \left(U_{i,j}^{n-\frac{1}{2}} + U_{i,j}^{n+\frac{1}{2}} \right) \Bigg) \\ + \Bigg(\left(V_{i+\frac{1}{2},j+\frac{1}{2}}^{n-\frac{1}{2}} + V_{i+\frac{1}{2},j+\frac{1}{2}}^{n+\frac{1}{2}} \right) - \left(V_{i+\frac{1}{2},j-\frac{1}{2}}^{n-\frac{1}{2}} + V_{i+\frac{1}{2},j-\frac{1}{2}}^{n+\frac{1}{2}} \right) \Bigg) \Bigg] \Delta x \Delta z \Delta t \end{aligned} \tag{27}$$

and for μ we get

$$\begin{aligned}\frac{\partial\chi}{\partial\mu} \approx \sum_{n=0}^{N_t-1}\sum_{i=0}^{N_x-1}\sum_{j=0}^{N_z-1}\Bigg\{ & -\frac{1}{\Delta x}\overline{X}^n_{i+\frac{1}{2},j}\left[\left(U^{n-\frac{1}{2}}_{i+1,j}+U^{n+\frac{1}{2}}_{i+1,j}\right)-\left(U^{n-\frac{1}{2}}_{i,j}+U^{n+\frac{1}{2}}_{i,j}\right)\right] \\ & -\frac{1}{\Delta z}\overline{Z}^n_{i+\frac{1}{2},j}\left[\left(V^{n-\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2}}+V^{n+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2}}\right)-\left(V^{n-\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2}}+V^{n+\frac{1}{2}}_{i+\frac{1}{2},j-\frac{1}{2}}\right)\right] \\ & -\frac{1}{2\Delta z}\overline{T}^n_{i,j+\frac{1}{2}}\left[\left(U^{n+\frac{1}{2}}_{i,j+1}+U^{n-\frac{1}{2}}_{i,j+1}\right)-\left(U^{n+\frac{1}{2}}_{i,j}+U^{n-\frac{1}{2}}_{i,j}\right)\right] \\ & -\frac{1}{2\Delta x}\overline{T}^n_{i,j+\frac{1}{2}}\left[\left(V^{n+\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2}}+V^{n-\frac{1}{2}}_{i+\frac{1}{2},j+\frac{1}{2}}\right)-\left(V^{n+\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2}}+V^{n-\frac{1}{2}}_{i-\frac{1}{2},j+\frac{1}{2}}\right)\right]\Bigg\}\Delta x\Delta z\Delta t\end{aligned} \tag{28}$$

where again we interpolated in time by averaging a field K at $K^n_{i,j} \approx \frac{1}{2}\left(K^{n+\frac{1}{2}}_{i,j}+K^{n-\frac{1}{2}}_{i,j}\right)$.

4.3 Model updating using L-BFGS gradient descent

With the gradients in hand, the most simple way to update the models in the gradient direction would just be at each iteration descend in the negative gradient direction.

$$\begin{cases}\vec{\rho}_k = \vec{\rho}_{k-1} - \alpha^\rho_k\gamma_\rho \\ \vec{\lambda}_k = \vec{\lambda}_{k-1} - \alpha^\lambda_k\gamma_\lambda \\ \vec{\mu}_k = \vec{\mu}_{k-1} - \alpha^\mu_k\gamma_\mu\end{cases} \tag{29}$$

where the index k denotes the iteration number, $\alpha^\rho_k, \alpha^\lambda_k, \alpha^\mu_k$ are step dependent positive stepsizes and the gradients $\vec{\gamma_m} = \frac{\partial\chi}{\partial\vec{m}}\ \forall\ \vec{m} \in \left\{\vec{\rho},\vec{\mu},\vec{\lambda}\right\}$. In practice, with simple gradient descend, these are constant during the iterations and are determined by hand by monitoring the cost function until it decreases sufficiently each iteration.

Now *Newton's method* for finding roots of function is often used in optimization due to it's incorporation of curvature information through the second derivative(or Hessian Matrix H in multi-dimensional problems). Newton's method improves on Eq. (29) by adding this Hessian information in the gradient descend according to

$$\begin{aligned}\vec{\rho}_k &= \vec{\rho}_{k-1} - \alpha^\rho_k H^{-1}_{\rho_k}\gamma_\rho \\ \vec{\lambda}_k &= \vec{\lambda}_{k-1} - \alpha^\lambda_k H^{-1}_{\lambda_k}\gamma_\lambda \\ \vec{\mu}_k &= \vec{\mu}_{k-1} - \alpha^\mu_k H^{-1}_{\mu_k}\gamma_\mu\end{aligned} \tag{30}$$

The inverse Hessian matrix H^{-1} is computationally expensive to calculate, which is why *quasi-Newton methods* were introduced, approximating the Hessian. One particular widely used version is the Broyden–Fletcher–Goldfarb–Shannon (BFGS) algorithm and it's limited memory variant L-BFGS [12]. This tries to approximate the product of the inverse Hessian and gradient by using information of the previous gradient descend steps(and thus requiring one to save the last L previous models and gradients costing extra memory). In this work we used a pre-written standalone module in C [13] which we fed the calculated gradients and models each iteration and which returned the product $H^{-1}\frac{\partial\chi}{\partial\vec{m}}$ for each parameter separately.

4.4 Implementation in C

The FWI inversion for the elastic wave equation was implemented in C in combination with Madagascar[14] for reproducible workflow, Jupyter-notebook[15] for data analysis and plotting and is available on request.[1].

4.4.1 Memory considerations

The forward modeling of the elastic wave equation in principle only requires one to store 14 $N_x \times N_z$ fields (3 models, 5 real fields and 6 CPML memory variables, which could be optimized to be only the size of the CPML layer at the cost of code clarity). And at each timestep the memory can be overwritten. For the adjoint back-propagation we have the same and thus need to store 13 more fields(5 + 8 CPML). The problem with gradients is that we need to calculate the product of the adjoint and forward fields at all timesteps. We thus need to store the time-slices of at least one of the two. We therefore stored U and V as 3D fields of size $N_x \times N_z \times N_t$ and calculated the gradient during the adjoint propagation loop (approximating $D_tU, D_tV, D_xU, D_xV, D_zU$ and D_zV during the adjoint loop instead of saving these 6 fields in the forward propagation). This provided the best trade-off between memory and computational cost, since memory was a real issue on the 16Gb RAM PC. The full algorithm is shown in Algorithm 3

[1] contact: Jaapwesdorp@gmail.com

Algorithm 3 FWI sequence

$d_U^{meas}, d_V^{meas} \leftarrow$ READ MEASURED SHOTS( ) ▷ Read measured shot data at all receiver positions
$\rho, \mu, \lambda \leftarrow$ READ INITIAL MODELS( )

for $k = 0$ to $k < N_{fwi-iterations}$ **do**
  for each source f_q **do** ▷ Loop over source positions(in parallel)
    $U, V \leftarrow$ PROPAGATE FORWARD(ρ, μ, λ) ▷ Save the 3D forward velocities for gradients
    $d_U^{mod}, d_V^{mod} \leftarrow$ SAVE MODELED SHOTS(U, V) ▷ Save shots at all receiver positions
    $\gamma_\rho, \gamma_\mu, \gamma_\lambda \leftarrow$ PROPAGATE ADJOINT AND CALCULATE GRADIENTS(U, V, ρ, μ, λ) ▷ Algorithm 2
  end for
  if $k < n_{\text{gradient descend}}$ **then** ▷ Perform initial gradient descend steps
    $\rho \leftarrow \rho - \alpha_k^\rho \gamma_\rho$ ▷ Stepsize α determined by hand
    $\mu \leftarrow \mu - \alpha_k^\mu \gamma_\mu$
    $\lambda \leftarrow \lambda - \alpha_k^\lambda \gamma_\lambda$
  else ▷ Perform L-BFGS steps
    $c_k \leftarrow 0$ ▷ Initialize cost function
    $\alpha_k^\rho, \alpha_k^\mu, \alpha_k^\mu \leftarrow 1$ ▷ Set initial L-BFGS stepsize to 1
    $\tilde{H}_\rho^{-1}\gamma_\rho, \tilde{H}_\mu^{-1}\gamma_\mu, \tilde{H}_\lambda^{-1}\gamma_\lambda \leftarrow$ APPROXIMATE HESSIANS VIA L-BFGS($k, \rho, \mu, \lambda, \gamma_\rho, \gamma_\mu, \gamma_\lambda$)
    $\rho \leftarrow \rho - \alpha_k^\rho \tilde{H}_\rho^{-1}\gamma_\rho$
    $\mu \leftarrow \mu - \alpha_k^\mu \tilde{H}_\mu^{-1}\gamma_\mu$
    $\lambda \leftarrow \lambda - \alpha_k^\lambda \tilde{H}_\lambda^{-1}\gamma_\lambda$
    for each source f_q **do**
      $U, V \leftarrow$ PROPAGATE FORWARD(ρ, μ, λ) ▷ Redo the forward modeling
      $d_U^{mod}, d_V^{mod} \leftarrow$ SAVE MODELED SHOTS(U, V)
      $c_k \leftarrow$ CALCULATE COST FUNCTION($d_U^{meas}, d_V^{meas}, d_U^{mod}, d_V^{mod}$) ▷ Eq. (4)
    end for
    while $c_k > c_{k-1}$ **do** ▷ Enter L-BFGS attempt loop if cost is not decreasing
      $\alpha_k^\rho \leftarrow \frac{1}{2}\alpha_k^\rho,\ \alpha_k^\mu \leftarrow \frac{1}{2}\alpha_k^\mu,\ \alpha_k^\lambda \leftarrow \frac{1}{2}\alpha_k^\lambda$
      $\rho \leftarrow \rho + \alpha_k^\rho \tilde{H}_\rho^{-1}\gamma_\rho$ ▷ Add back half the step taken
      $\mu \leftarrow \mu + \alpha_k^\mu \tilde{H}_\mu^{-1}\gamma_\mu$
      $\lambda \leftarrow \lambda + \alpha_k^\lambda \tilde{H}_\lambda^{-1}\gamma_\lambda$
      for each source f_q **do**
        $U, V \leftarrow$ PROPAGATE FORWARD(ρ, μ, λ) ▷ Redo the forward modeling
        $d_U^{mod}, d_V^{mod} \leftarrow$ SAVE MODELED SHOTS(U, V)
        $c_k \leftarrow$ CALCULATE COST FUNCTION($d_U^{meas}, d_V^{meas}, d_U^{mod}, d_V^{mod}$)
      end for
    end while
  end if
end for

5. FWI results

FWI was tested on a synthetic model (Section 5.1) with both larger features (the CPS letters) and smaller (research group). The grid size was always $N_x = 113$ by $N_z = 214$ points. The grid-spacing was taken to be $\Delta_x = \Delta_z = 10$m and Brocher's relations Eqs. (21) and (22) were used to calculate v_s and ρ from v_p. The central source frequency was 3Hz (resulting in 54 Points per wavelength (PPW) for P-waves and 14 for S-waves) and the inversion was carried out for 7 sources in parallel denoted by the blue dots in Section 5.1 and receivers at every 10m denoted by the red line.

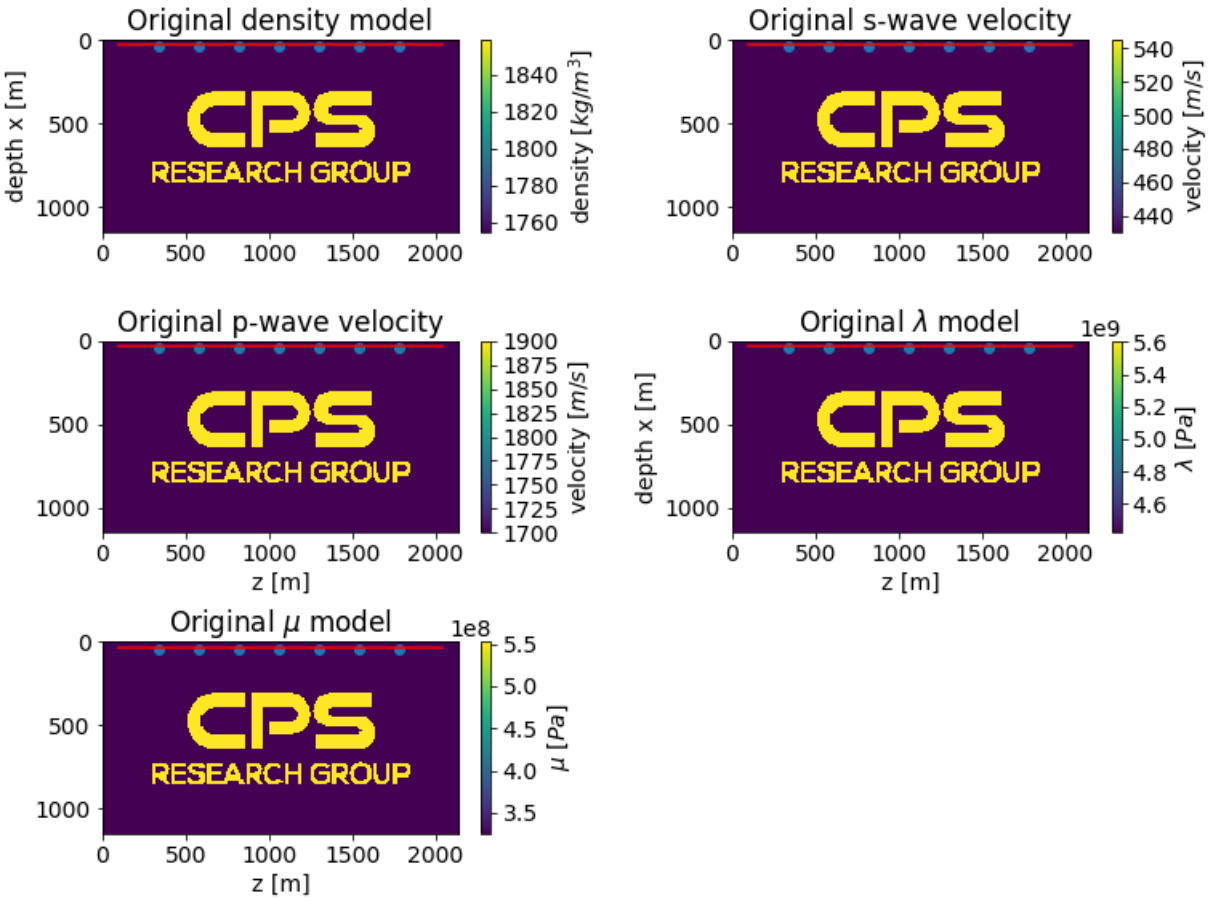


Figure 4. Synthetic models used for testing FWI

For the FWI we took 2 standard gradient descend steps before applying L-BFGS. The initial stepsize α was determined by hand for each parameter separately by looking at the first step and taking the value where the cost was sufficiently decreasing. As can be seen later, the range in which this parameter can be chosen for the L-BFGS to be effective spans several orders of magnitude.

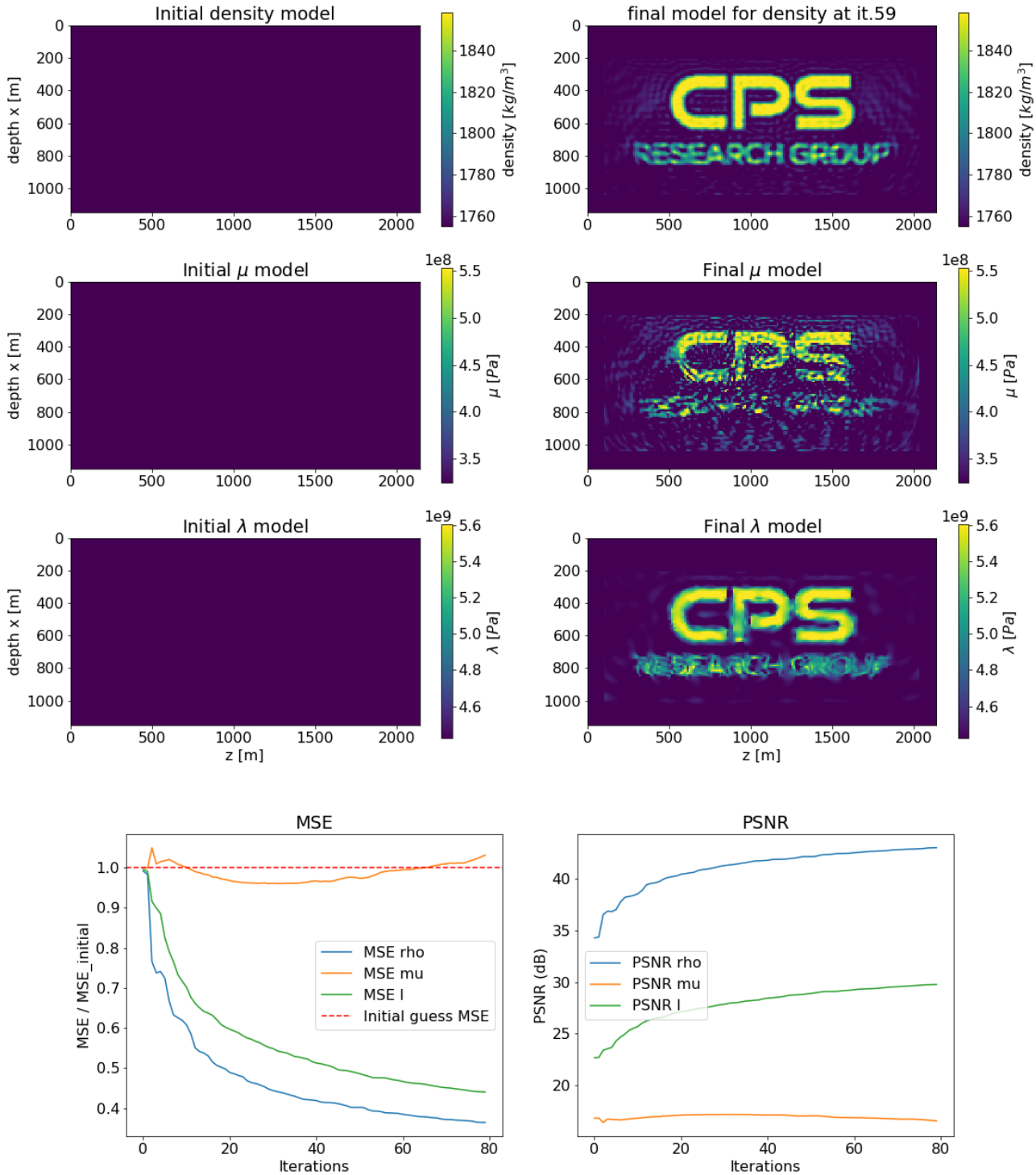


Figure 5. One by one sequential single parameter inversion with an empty homogeneous starting model with the other two parameters held fixed on the original and the original taken as Section 5.1

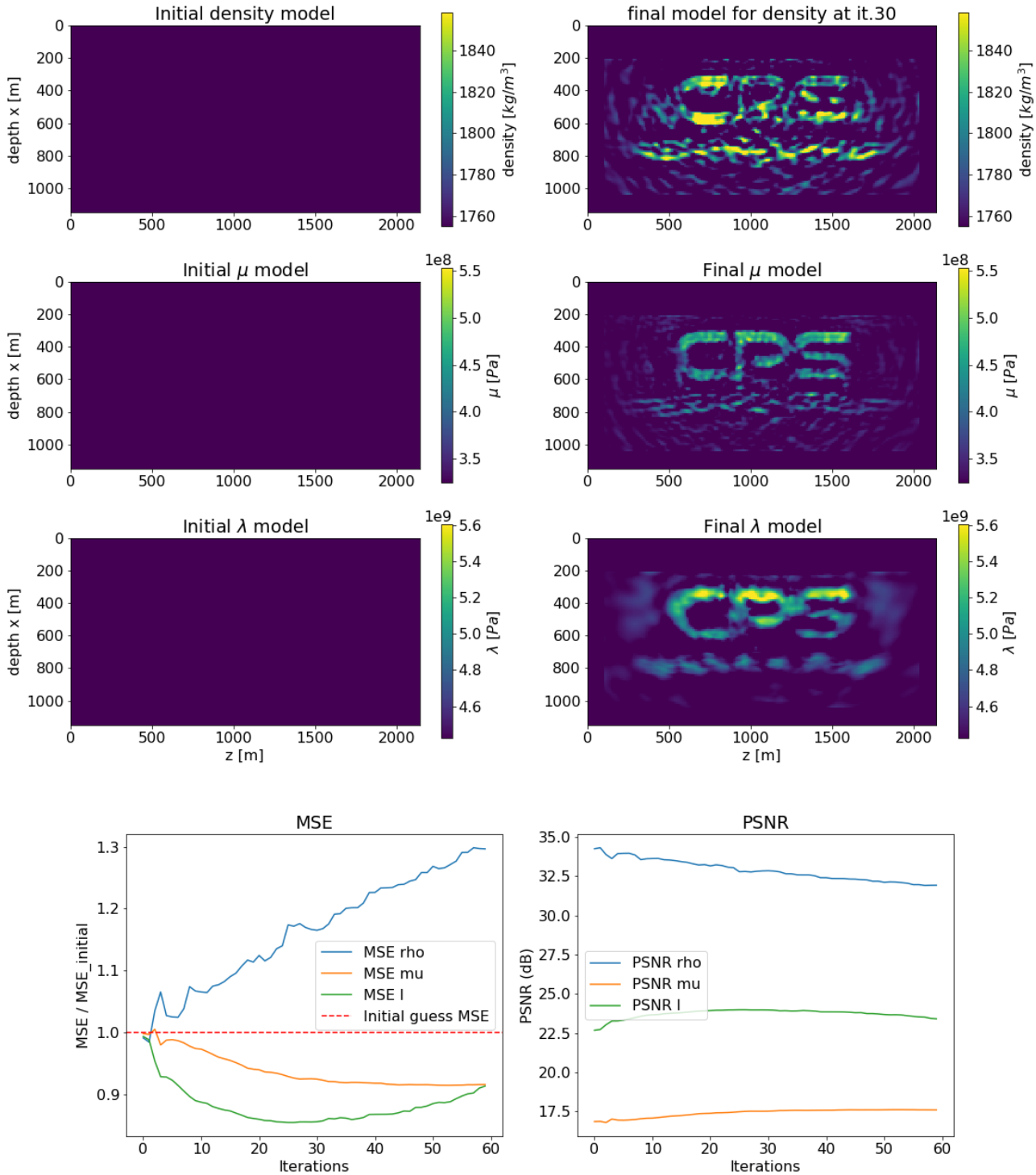


Figure 6. Three parameter inversion result with a homogeneous starting model and the original taken as Section 5.1

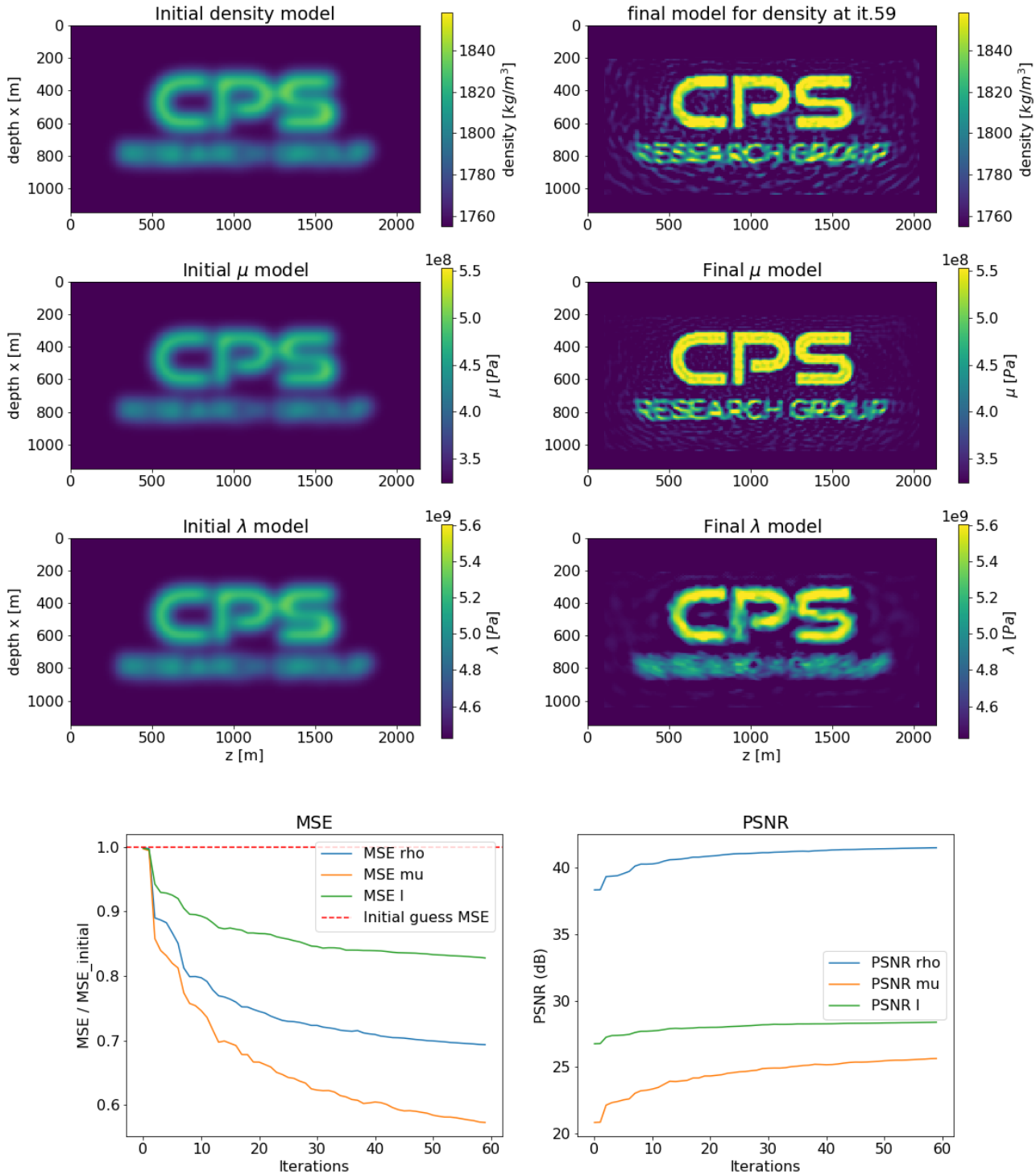


Figure 7. Three parameter inversion result with a smooth starting model and the original taken as Section 5.1

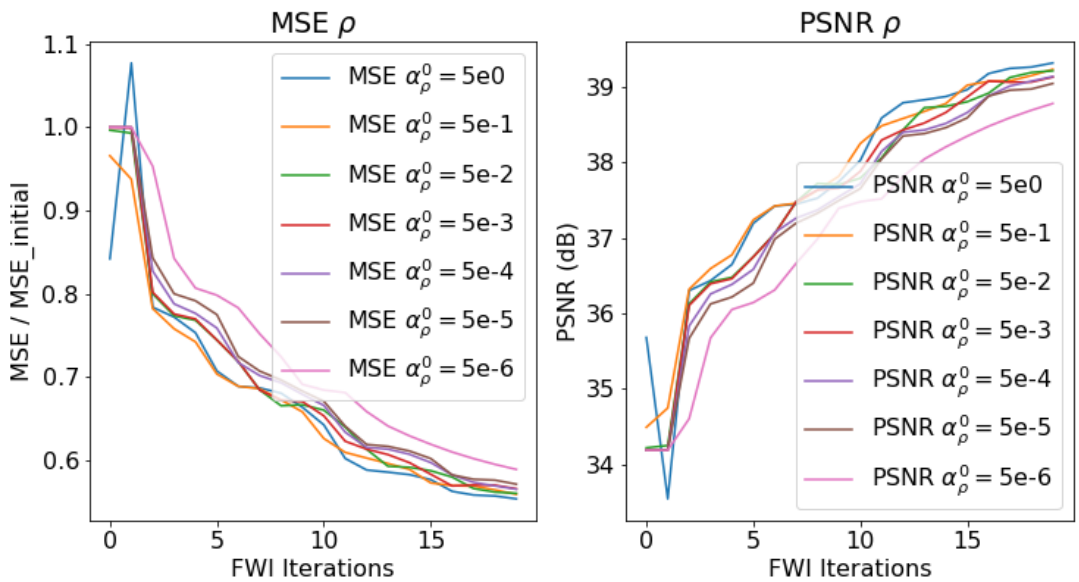


Figure 8. Comparison of final errors and PSNR for different initial stepsizes.

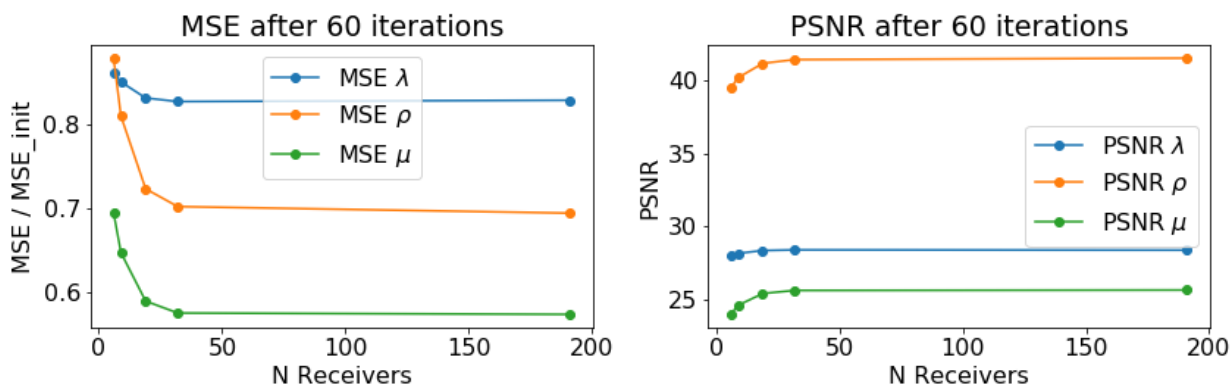


Figure 9. Final reconstruction quality with varying amount of receivers. The reconstruction was performed with the smooth starting model from Section 5.1 for varying $N_R \in \{191, 32, 19, 9, 6\}$

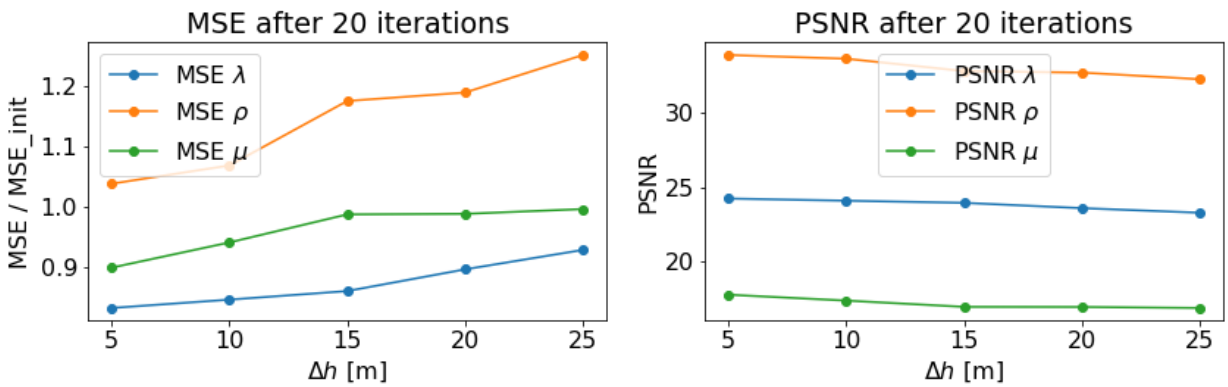


Figure 10. Final reconstruction quality of CPS model with homogeneous starting model with varying gridsize $\Delta h = \Delta x = \Delta z$ the propagation time was increased with gridsize linearly to make sure the wave always propagated the same distance. As gridsize increases, the amount of PPW decreases and therefore the reconstruction quality. Because the amount of points were fixed due to memory limitations, the physical size of the letters increased with Δh which in prinicple should increase the reconstruction quality.

5.1 Rescaled version of realistic seismic testing model

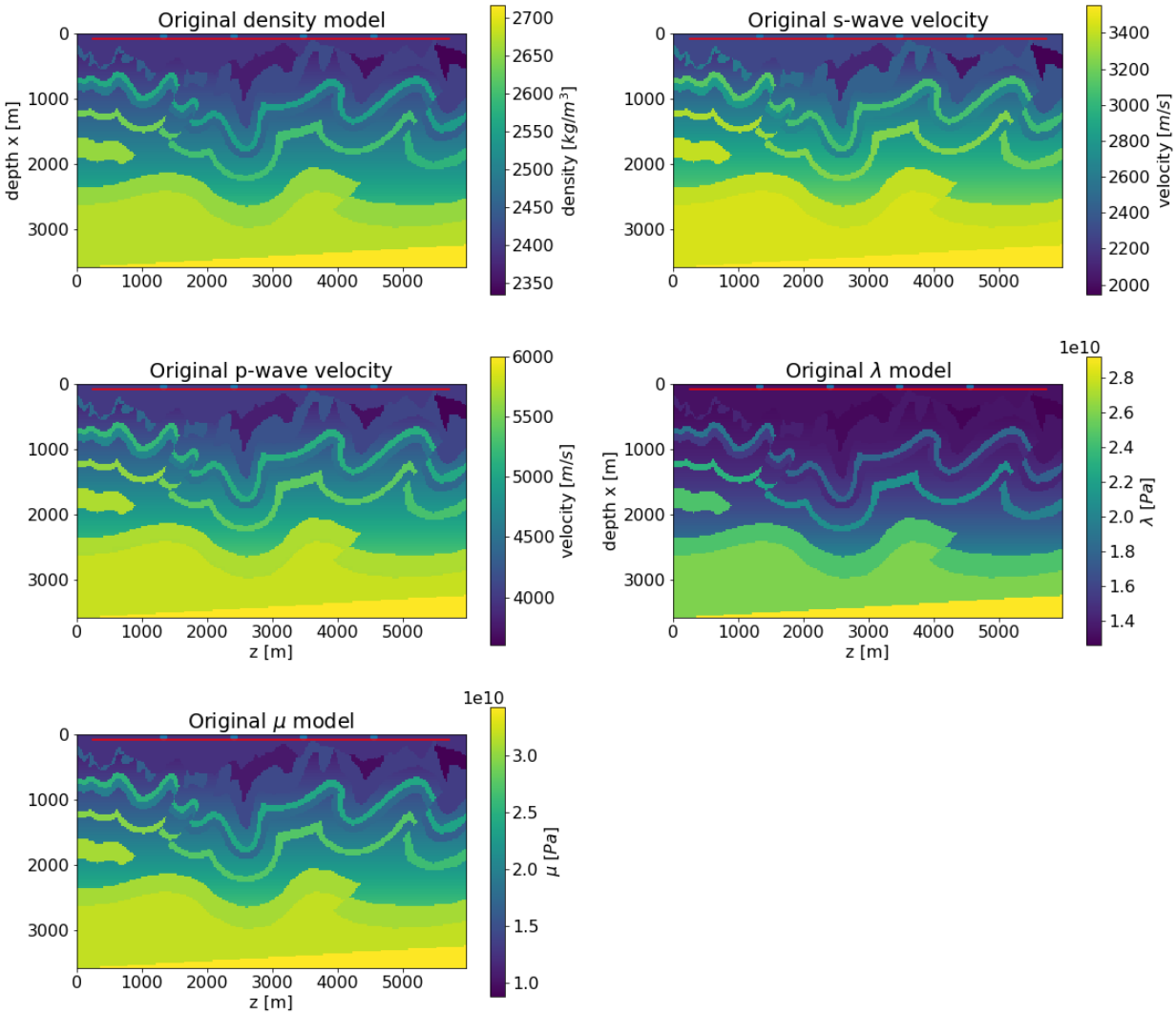


Figure 11. Realistic synthetic models [16]. The model was rescaled by about a factor 3.5 from the original as to fit in a $\Delta x = \Delta z = 25$m with only 243 by 143 grid points.

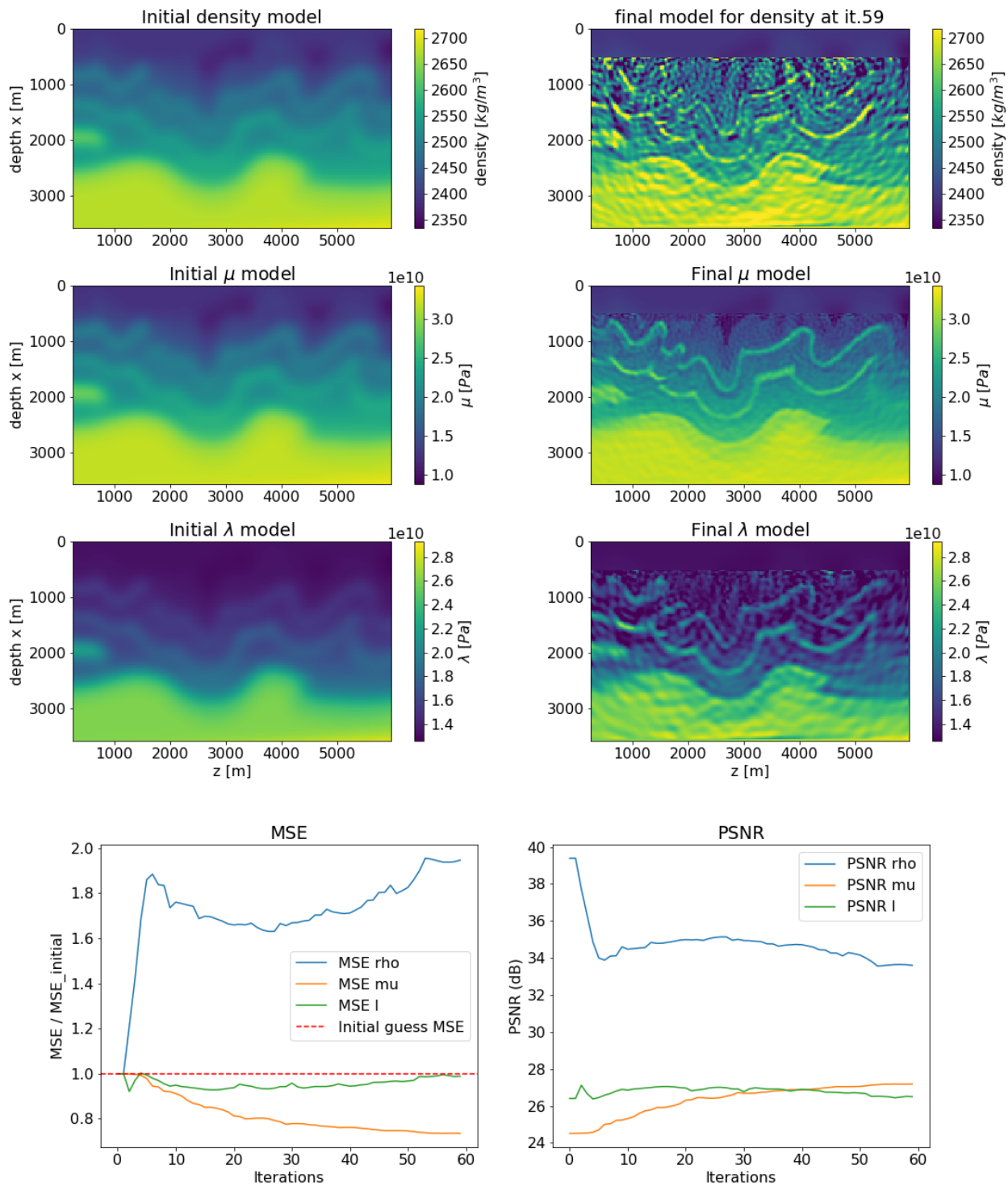


Figure 12. Three parameter inversion result with a smooth starting model and the original taken as Section 5.1

6. Conclusion

Appendices

1. Lagrangian based derivation of the gradient via the adjoint state method

In this section the adjoint equations and the gradients for a general first or second order PDE system will be derived performing a more generalized version of the approach outlined in [3].

A.1 General system of equations

We can write the most general minimization problem of a cost function subject any first order or second order (in time) set of partial differential equations(PDE) in the following form

$$\begin{aligned} \min_{m} \quad & \chi(s,m) = \int_0^T f(s,m)\,dt \\ \text{subject to} \quad & h(\ddot{s},\dot{s},s,m,t) = 0 \\ \text{with B.C} \quad & g((s(0),m) = 0 \\ & k((\dot{s}(0),m) = 0 \end{aligned} \tag{31}$$

where s is a discretized vector of the fields and the dots denote derivatives to time, m is the vector of all the discretized model parameters, t denotes time, $h(\ddot{s},\dot{s},s,m,t) = 0$ is the system of equations, e.g the wave equation in 2D or 3D,T is the final time of integration and $g((s(0),m),k((\dot{s}(0),m)$ denote initial conditions for the field vector s and its derivative $\dot{s}$. Note that this is similar to [3] but with the added explicit dependence of both $\dot{s}$ and $\ddot{s}$ giving a more general expression for the adjoint equations and the gradients.

A.2 Derivative to model parameters

When solving the minimization problem numerically, one often uses a method similar to Newton's gradient descend. This thus requires knowledge of the derivative of the cost function $\chi(s,m)$ to all of the model parameters m. Using the chain rule we obtain:

$$\frac{d\chi}{dm} = \frac{\partial\chi}{\partial s}\frac{\partial s}{\partial m} + \frac{\partial\chi}{\partial m} \tag{32}$$

This depends on the Frechet derivatives $\frac{\partial s}{\partial m}$ which require at least $3N$ evaluations of the forward model in order to obtain an estimate, we thus want to avoid the calculation of this term. To facilitate this we define the Lagrangian $\mathscr{L}$ by

$$\mathscr{L} = \int_0^T \left[f(s,m) + \lambda^T h(\ddot{s},\dot{s},s,m,t)\right] dt + \mu^T g((s(0),m) + \eta^T k((\dot{s}(0),m) \tag{33}$$

where the auxiliary variables λ, μ, η have the same length as the discretized field vector s. Note that due to the initial conditions of Eq. (31) we have $\frac{d\chi}{dm} = \frac{d\mathscr{L}}{dm}$, where using the chain rule repetitively

$$\begin{aligned}\frac{d\mathscr{L}}{dm} = \int_0^T \Bigg[& \frac{\partial f}{\partial s}\frac{\partial s}{\partial m} + \frac{\partial f}{\partial m} \\ & + \lambda^T \left(\frac{\partial h}{\partial \ddot{s}}\frac{\partial \ddot{s}}{\partial m} + \frac{\partial h}{\partial \dot{s}}\frac{\partial \dot{s}}{\partial m} + \frac{\partial h}{\partial s}\frac{\partial s}{\partial m} + \frac{\partial h}{\partial m} \right) \Bigg] dt \\ & + \mu^T \left(\frac{\partial g}{\partial s(0)}\frac{\partial s(0)}{\partial m} + \frac{\partial g}{\partial m} \right) \\ & + \eta^T \left(\frac{\partial k}{\partial \dot{s}(0)}\frac{\partial \dot{s}(0)}{\partial m} + \frac{\partial k}{\partial m} \right)\end{aligned} \tag{34}$$

This looks intimidating and not really much simpler, but we are still free to choose the expressions for λ, μ and η. We will choose this such that we can avoid calculating the computationally difficult derivatives $\frac{\partial s}{\partial m}$. But first we need to do some partial integration to get rid of $\frac{\partial \dot{s}}{\partial m}$ and $\frac{\partial \ddot{s}}{\partial m}$. With a single partial integration we can write

$$\int_0^T \lambda^T \frac{\partial h}{\partial \dot{s}}\frac{\partial \dot{s}}{\partial m} = \left[\lambda^T \frac{\partial h}{\partial \dot{s}}\frac{\partial s}{\partial m} \right]_0^T - \int_0^T \frac{\partial s}{\partial m} \left(\dot{\lambda}^T \frac{\partial h}{\partial \dot{s}} + \lambda^T \frac{\partial}{\partial t}\frac{\partial h}{\partial \dot{s}} \right) \tag{35}$$

and with a double partial integration we can write

$$\begin{aligned}\int_0^T \lambda^T \frac{\partial h}{\partial \ddot{s}}\frac{\partial \ddot{s}}{\partial m} = & \left[\lambda^T \frac{\partial h}{\partial \dot{s}}\frac{\partial \dot{s}}{\partial m} - \frac{\partial s}{\partial m} \left(\dot{\lambda}^T \frac{\partial h}{\partial \ddot{s}} + \lambda^T \frac{\partial}{\partial t}\frac{\partial h}{\partial \ddot{s}} \right) \right]_0^T \\ & + \int_0^T \frac{\partial s}{\partial m} \left(\ddot{\lambda}^T \frac{\partial h}{\partial \ddot{s}} + 2\dot{\lambda}^T \frac{\partial}{\partial t}\frac{\partial h}{\partial \ddot{s}} + \lambda^T \frac{\partial^2}{\partial t^2}\frac{\partial h}{\partial \ddot{s}} \right) dt\end{aligned} \tag{36}$$

So filling this back into Eq. (34) and regrouping terms gives

$$\begin{aligned}\frac{d\mathscr{L}}{dm} = \int_0^T \Bigg[& \frac{\partial s}{\partial m} \left(\frac{\partial f}{\partial s} + \ddot{\lambda}^T \frac{\partial h}{\partial \ddot{s}} + \dot{\lambda}^T \left(2\frac{\partial}{\partial t}\frac{\partial h}{\partial \ddot{s}} - \frac{\partial h}{\partial \dot{s}} \right) \right. \\ & \left. + \lambda^T \left(\frac{\partial h}{\partial s} + \frac{\partial^2}{\partial t^2}\frac{\partial h}{\partial \ddot{s}} - \frac{\partial}{\partial t}\frac{\partial h}{\partial \dot{s}} \right) \right) + \frac{\partial f}{\partial m} + \lambda^T \frac{\partial h}{\partial m} \Bigg] dt \\ & + \left(\mu^T \frac{\partial g}{\partial s(0)} - \lambda^T \frac{\partial h}{\partial \dot{s}} + \dot{\lambda}^T \frac{\partial h}{\partial \ddot{s}} + \lambda^T \frac{\partial}{\partial t}\frac{\partial h}{\partial \ddot{s}} \right) \frac{\partial s}{\partial m}\bigg|_0 \\ & + \left(\eta^T \frac{\partial h}{\partial \dot{s}(0)} - \lambda^T \frac{\partial h}{\partial \ddot{s}} \right) \frac{\partial \dot{s}}{\partial m}\bigg|_0 \\ & + \lambda^T \frac{\partial h}{\partial \ddot{s}}\frac{\partial \dot{s}}{\partial m}\bigg|_T + \left(\lambda^T \frac{\partial h}{\partial \dot{s}} - \dot{\lambda}^T \frac{\partial h}{\partial \ddot{s}} - \lambda^T \frac{\partial}{\partial t}\frac{\partial h}{\partial \ddot{s}} \right) \frac{\partial s}{\partial m}\bigg|_T \\ & + \mu^T \frac{\partial g}{\partial m} + \eta^T \frac{\partial k}{\partial m}\end{aligned} \tag{37}$$

we can then choose values for the auxiliary variables such that undesired terms drop out of $\frac{d\mathscr{L}}{dm}$. If we set

$$\begin{aligned}
\mu^T &= \left(\dot{\lambda}(0)\frac{\partial h}{\partial \ddot{s}} + \lambda^T(0)\left(\frac{\partial}{\partial t}\frac{\partial h}{\partial \ddot{s}} - \frac{\partial h}{\partial \dot{s}}\right)\right)\left(\frac{\partial g}{\partial s(0)}\right)^{-1} \\
\eta^T &= \left(\lambda(0)\frac{\partial h}{\partial \ddot{s}}\right)\left(\frac{\partial k}{\partial \dot{s}(0)}\right)^{-1} \\
\lambda(T) &= 0 \\
\dot{\lambda}(T) &= 0
\end{aligned} \tag{38}$$

and let λ satisfy the following so called *adjoint* equation

$$\ddot{\lambda}^T\frac{\partial h}{\partial \ddot{s}} + \dot{\lambda}^T\left(2\frac{\partial}{\partial t}\frac{\partial h}{\partial \ddot{s}} - \frac{\partial h}{\partial \dot{s}}\right) + \lambda^T\left(\frac{\partial h}{\partial s} + \frac{\partial^2}{\partial t^2}\frac{\partial h}{\partial \ddot{s}} - \frac{\partial}{\partial t}\frac{\partial h}{\partial \dot{s}}\right) = -\frac{\partial f}{\partial s} \tag{39}$$

we can see that the $\frac{\partial s}{\partial m}$ terms drop out of $\frac{d\mathscr{L}}{m}$ and we remain for the *gradients* only with

$$\frac{d\mathscr{L}}{dm} = \int_0^T \frac{\partial f}{\partial m} + \lambda^T\frac{\partial h}{\partial m}dt + \mu^T\frac{\partial g}{\partial m} + \eta^T\frac{\partial k}{\partial m} \tag{40}$$

where the two rightmost terms are zero if the initial conditions do not depend directly on the model parameters m. Eq. (39) is called the *adjoint* equation to the original system of equations $h(\ddot{s},\dot{s},s,m,t)$

2. More example applications

B.1 The second order electromagnetic wave equation

Now for an example with second order terms as well we take the following second order electromagnetic wave equation

$$\varepsilon\frac{\partial^2 E_y}{\partial t^2} = \frac{\partial}{\partial x}\left(\frac{1}{\mu}\frac{\partial E_y}{\partial x}\right) + \frac{\partial}{\partial z}\left(\frac{1}{\mu}\frac{\partial E_y}{\partial z}\right) - \sigma\frac{\partial E_y}{\partial t} \tag{41}$$

which models the same as **??**. We can see that now $s = \vec{E}_y$ a single field variable (which is still a vector of length N)

$$T(m) = \varepsilon,\ C(m) = -\sigma,\ A(m) = \mathscr{D}_x\frac{1}{\mu}\mathscr{D}_x + \mathscr{D}_z\frac{1}{\mu}\mathscr{D}_z \tag{42}$$

All of these operators are self-adjoint, thus $A^T(m) = A(m),\ C^T(m) = C(m),\ T^T(m) = T(m)$, we thus get for the adjoint equation

$$\varepsilon\frac{\partial^2 \lambda}{\partial t^2} = \frac{\partial}{\partial x}\left(\frac{1}{\mu}\frac{\partial \lambda}{\partial x}\right) + \frac{\partial}{\partial z}\left(\frac{1}{\mu}\frac{\partial \lambda}{\partial z}\right) + \sigma\frac{\partial \lambda}{\partial t} \tag{43}$$

where there is only a sign change in front of σ due to Eq. (3). The gradients then become according to Eq. (2)

$$\begin{aligned}
\frac{d\chi}{d\varepsilon} &= \int_0^T \vec{\lambda}^T\ddot{\vec{E}}_y\,dt \\
\frac{d\chi}{d\mu} &= \int_0^T \vec{\lambda}^T\left(\frac{\partial}{\partial x}\left(\frac{1}{\mu^2}\frac{\partial \vec{E}_y}{\partial x}\right) + \frac{\partial}{\partial z}\left(\frac{1}{\mu^2}\frac{\partial \vec{E}_y}{\partial z}\right)\right)dt \\
\frac{d\chi}{d\sigma} &= \int_0^T \vec{\lambda}^T\dot{\vec{E}}_y\,dt
\end{aligned} \tag{44}$$

B.2 The second order acoustic wave equation

The acoustic wave equation with variable density ρ and velocity c is given by

$$\frac{1}{\rho c^2}\frac{\partial^2 p}{\partial t^2} = \left(\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial p}{\partial x}\right) + \frac{\partial}{\partial z}\left(\frac{1}{\rho}\frac{\partial p}{\partial z}\right)\right) \tag{45}$$

we thus get

$$T(m) = \frac{1}{\rho c^2},\ C(m) = 0,\ A(m) = \mathscr{D}_x\frac{1}{\rho}\mathscr{D}_x + \mathscr{D}_z\frac{1}{\rho}\mathscr{D}_z \tag{46}$$

which looks very similar to the electromagnetic case, again $A^T(m) = A(m),\ C^T(m) = C(m),\ T^T(m) = T(m)$, so following Eq. (3) the adjoint equation is the same as the original equation

$$\frac{1}{\rho c^2}\frac{\partial^2 \lambda}{\partial t^2} = \left(\frac{\partial}{\partial x}\left(\frac{1}{\rho}\frac{\partial \lambda}{\partial x}\right) + \frac{\partial}{\partial z}\left(\frac{1}{\rho}\frac{\partial \lambda}{\partial z}\right)\right) \tag{47}$$

According to Eq. (2) we then obtain

$$\begin{aligned}\frac{d\chi}{dc} &= -\frac{2}{\rho c^3}\int_0^T \vec{\lambda}^T \ddot{\vec{p}}\, dt \\ \frac{d\chi}{d\rho} &= \int_0^T \vec{\lambda}^T \left(-\frac{1}{\rho^2 c^2}\ddot{\vec{p}} + \frac{\partial}{\partial x}\left(\frac{1}{\rho^2}\frac{\partial \vec{p}}{\partial x}\right) + \frac{\partial}{\partial z}\left(\frac{1}{\rho^2}\frac{\partial \vec{p}}{\partial z}\right)\right) dt\end{aligned} \tag{48}$$

B.3 The first order acoustic wave equation

The first order acoustic wave equation with variable density ρ and velocity c is given by

$$\begin{aligned}\frac{1}{\rho c^2}\frac{\partial p}{\partial t} &= -\frac{\partial v_x}{\partial x} - \frac{\partial v_z}{\partial z} \\ \frac{\partial v_x}{\partial t} &= -\frac{1}{\rho}\frac{\partial p}{\partial x} \\ \frac{\partial v_z}{\partial t} &= -\frac{1}{\rho}\frac{\partial p}{\partial z}\end{aligned} \tag{49}$$

this gives

$$T(m) = 0,\ C(m) = -\begin{bmatrix}\frac{1}{\rho c^2} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1\end{bmatrix},\ A(m) = \begin{bmatrix}0 & -\mathscr{D}_x & -\mathscr{D}_z \\ -\frac{1}{\rho}\mathscr{D}_x & 0 & 0 \\ -\frac{1}{\rho}\mathscr{D}_z & 0 & 0\end{bmatrix} \tag{50}$$

we thus get $C^T(m) = C$ and

$$A^T(m) = \begin{bmatrix}0 & \mathscr{D}_x\frac{1}{\rho} & \mathscr{D}_z\frac{1}{\rho} \\ \mathscr{D}_x & 0 & 0 \\ \mathscr{D}_z & 0 & 0\end{bmatrix} \tag{51}$$

so the adjoint equations according to Eq. (3) become

$$\begin{aligned}\frac{1}{\rho c^2}\frac{\partial \lambda_1}{\partial t} &= -\frac{\partial\left(\frac{1}{\rho}\lambda_2\right)}{\partial x} - \frac{\partial\left(\frac{1}{\rho}\lambda_3\right)}{\partial z} \\ \frac{\partial \lambda_2}{\partial t} &= -\frac{\partial \lambda_1}{\partial x} \\ \frac{\partial \lambda_3}{\partial t} &= -\frac{\partial \lambda_1}{\partial z}\end{aligned} \tag{52}$$

and the gradients according to Eq. (2)

$$\begin{aligned}
\frac{d\chi}{dc} &= -\frac{2}{\rho c^3}\int_0^T \vec{\lambda_1}^{\,T}\vec{p}\,dt \\
\frac{d\chi}{d\rho} &= \int_0^T \frac{-1}{\rho^2 c^2}\vec{\lambda_1}^{\,T}\vec{p} - \frac{1}{\rho^2}\left(\vec{\lambda_2}^{\,T}\frac{\partial\vec{p}}{\partial x} + \vec{\lambda_3}^{\,T}\frac{\partial\vec{p}}{\partial z}\right)\,dt
\end{aligned} \tag{53}$$

3. Rayleigh wave free surface test

The free surface condition of Eq. (24) was tested on a two layer model and resulted in a nicely visible Rayleigh wave traveling to the right, together with P and S waves and their reflections on the boundary.

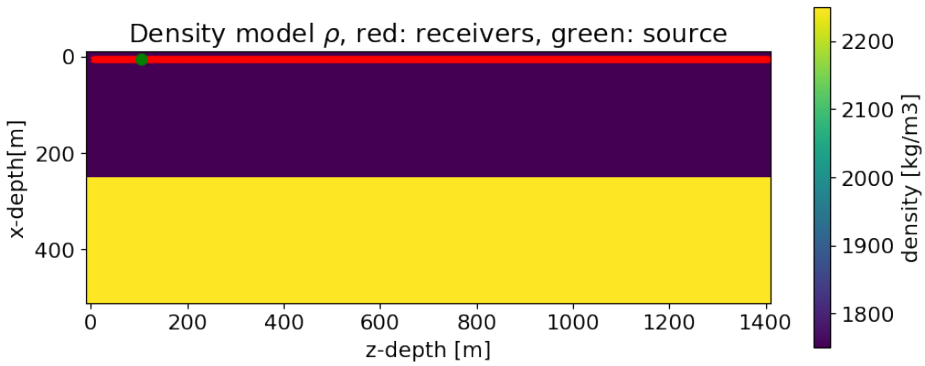


Figure 13. Earth model with 2 different materials. In the top layer v_p =1800 (m/s), v_s =1000 (m/s) and ρ =1750 (kg/m3). In the bottom layer v_p =3000 (m/s), v_s =1500 (m/s) and ρ =2250 (kg/m3).

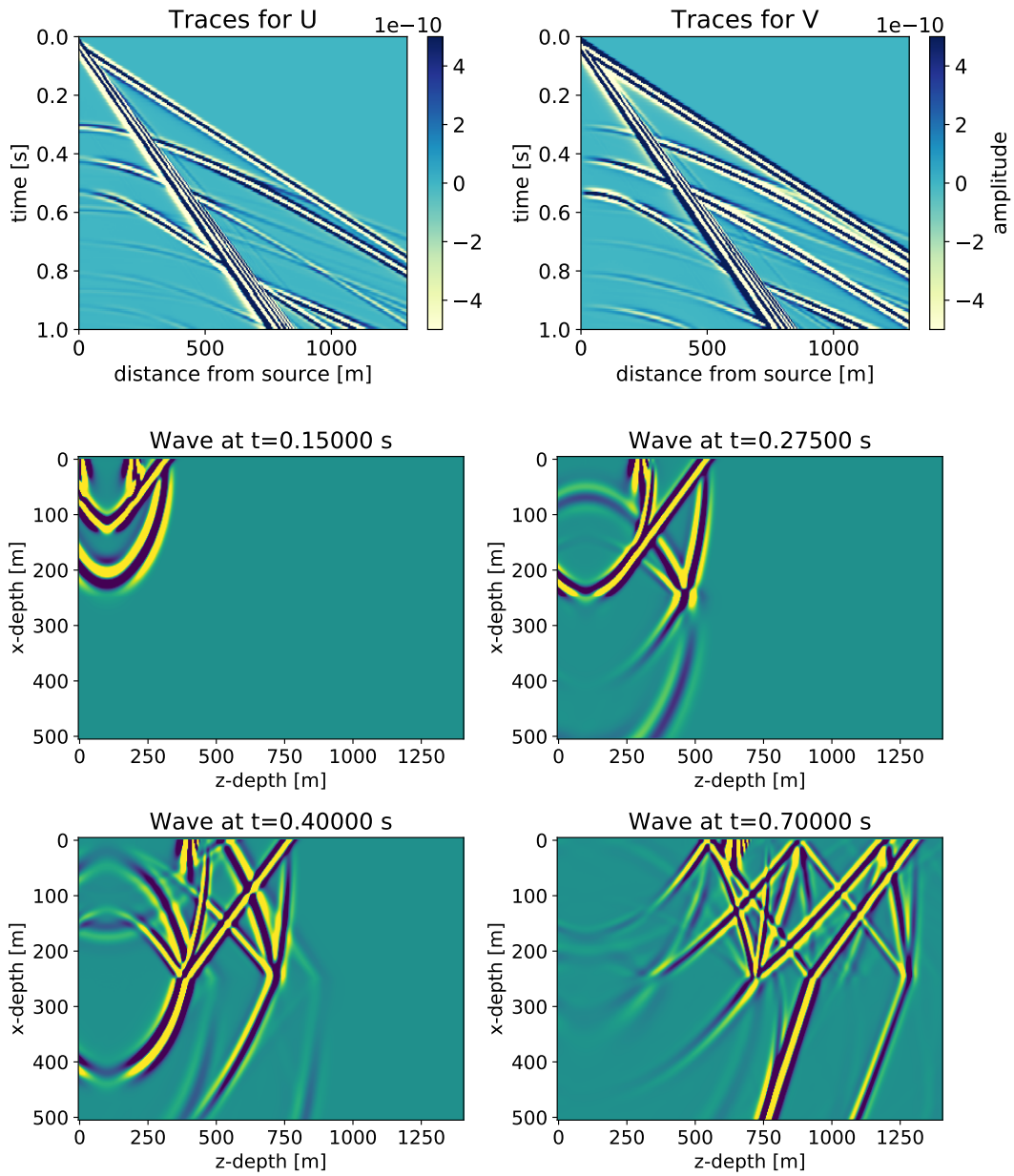


Figure 14. Free surface applied to the x=0 plane, top represent the shots and bottom four the V_x field slices. Total grid format 2800 by 1000 points and 20000 timesteps, $dx = dz = 0.5m$, $dt = 0.05$ms, $f_q = 30$Hz, CPML was applied on the other boundaries with standard settings(e.g 10 points, $R = 0.001$).

References

[1] Albert Tarantola. A strategy for nonlinear elastic inversion of seismic reflection data. *GEOPHYSICS*, 51(10):1893–1903, 1986.

[2] Rene Edouard Plessix. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophysical Journal International*, 167(2):495–503, 2006.

[3] Andrew M. Bradley. PDE-constrained optimization problems and the adjoint method. 2012(2):1–6, 2012.

[4] Jean Virieux. SH-wave propagation in heterogeneous media : Velocity-stress finite-difference method. *Geophysics*, 49(11):1933–1957, 1984.

[5] Jean-Pierre Berenger. A perfectly matched layer for the absorption of electromagnetic waves, 1994.

[6] J. Alan Roden and Stephen D. Gedney. Convolutional PML(C-PML): an efficient FDTD implementation of the C-PML for arbitrary media. *Microwave and Optical Technology Letters*, 27(5):334–339, 2000.

[7] Dimitri Komatitsch and Roland Martin. An unsplit convolutional perfectly matched layer improved at grazing incidence for the seismic wave equation. *Geophysics*, 72(5):SM155–SM167, 2007.

[8] Francis Collino and Chrysoula Tsogka. Application of the perfectly matched absorbing layer model to the linear elastodynamic problem in anisotropic heterogeneous media. *Geophysics*, 66(1):294–307, 2001.

[9] Thomas M. Brocher. Empirical relations between elastic wavespeeds and density in the Earth's crust. *Bulletin of the Seismological Society of America*, 95(6):2081–2092, 2005.

[10] Alan R. Levander. Fourth-order finite-difference P-5V seismograms. *Geophysics*, 53(11):1425–1436, 1988.

[11] Chong Zeng, Jianghai Xia, Richard D. Miller, and Georgios P. Tsoflias. An improved vacuum formulation for 2D finite-difference modeling of Rayleigh waves including surface topography and internal discontinuities. *Geophysics*, 77(1):T1–T9, 2012.

[12] Dong C. Liu and Jorge Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1-3):503–528, 1989.

[13] Sergio Alberto and Abreo Carrillo. *Tuning Up Global Optimization Techniques to Solve the Reliability Problem in Nonlinear Geophysical Inversions*. 2018.

[14] S. Fomel, P. Sava, I. Vlad, Y. Liu, and V. Bashkardin. Madagascar: open-source software project for multidimensional data analysis and reproducible computational experiments. *Journal of Open Research Software*, 1(1):e8, 2013.

[15] Thomas Kluyver, Benjamin Ragan-kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, Jessica Hamrick, Jason Grout, Sylvain Corlay, Paul Ivanov, Damián Avila, Safia Abdalla, and Carol Willing. Jupyter Notebooks—a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, pages 87–90, 2016.

[16] Samuel H Gray and Kurt J. Marfurt. Migration from topography: improving the near-surface image. *Canadian journal of exploration geophysics*, 31:18–24, 1995.