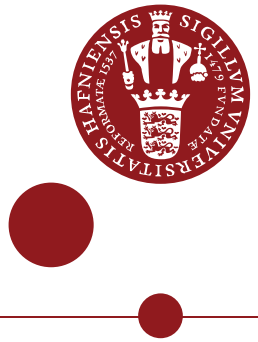
UNIVERSITY OF COPENHAGEN



MSc in Computational Physics

Deterministic wavelet estimation methods in the tie-to-well approach

Pedro Martínez Saiz

Supervised by Klaus Mosegaard

May 20th, 2022

Pedro Martínez Saiz

Deterministic wavelet estimation methods in the tie-to-well approach

MSc in Computational Physics, May 20th, 2022

Supervisor: Klaus Mosegaard

University of Copenhagen

Niels Bohr Institute

Section for the Physics of Ice, Climate and Earth

Tagensvej 16

2200 Copenhagen N

Acknowledgements

I cannot begin to express my thanks to Klaus, who has encouragingly guided me and introduced me to the world of seismology with this project. All the experience acquired along the way is always very satisfactory.

Very special thank you to my family, who, from the distance, has supported me in every single aspect and made this experience possible. Special mention to my little brother Hugo. Your endless child love knows no borders, you are my pillar and you do not know it yet.

Likewise, I could have not undertaken this journey without the family I found in Copenhagen. Lasse, Anne, Roosa, Alicia and many more: you are my motor and my energy. I am also very grateful to my MSc office mates for their moral support every day. Special mention to Martin, time flies and "we made it".

I would be remiss in not mentioning Teresa, with whom I started taking the first steps in Physics. I wish I could have you by my side in person and not on a screen.

Lastly, I very much appreciate the assistance of Niels. SPECFEM did not take over me, but the other way around thanks to him.

Abstract

Wavelets are wave-like functions with short extension, i.e., they oscillate over a short distance and damp very fast. Their main value over the whole domain is zero. These properties give wavelets the localization power, useful in a wide range of disciplines.

In seismology, the wavelets are hidden in the seismic data. Seismic processing methods are aimed at removing the wavelets from the seismic data to yield the response of the Earth to an initial perturbation: the reflectivity. The reflection coefficient series carry information about the structure of the subsurface. The aim of the project is to exploit deterministic inversion tools to estimate the wavelet function (our *unknown*) hidden in the data.

Deterministic inversion theory covers the those minimization problems that are solved using least squares approaches. During the project we will see that the least squares fit, in its simplest form, is not robust enough, and needs to be reformulated. More specifically, we will deploy Tikhonov regularization to render the problem robust and stable. Dumped Tikhonov regularization will be presented too as need of introducing prior information about the wavelets.

A forward model that parametrizes the problem is required. The convolutional model will be employed as the connecting bridge between the data space and the parameter space. Our parameters will be the wavelet itself, which must fulfil the foregoing requisites. We will explore its limitations, as well as up to which extent it works. We will simulate our own data using the SPECFEM software.

Contents

Chapter 1.

Introduction

Waves are our best tool to gain information of the material they travel through. It is well known that when a wave encounters a change in the material (interface), part of it is reflected back and the other part continues its course as the refracted wave. The refracted wave may meet further interfaces or reflectors, leading to multiple reflections that carry information about the structure of the Earth right beneath us. Consequently, reflection of waves plays a pivotal role when it comes to inferring the sub-surface structure. Some receivers are usually laid out along the surface waiting for signals from all these reflections to arrive. Yet, the reflections are not recorded directly, but encoded in the so-called seismic data, which are the Earth's response to a perturbation in form of wave, also referred to as seismograms or seismic traces. As a result, the wave field recorded is called *seismic section*, and, after some treatment, can be used as a rough "image" of the structure.

The aim is then twofold: collect the responses of seismic waves propagating through some media (forward simulations) and deploy some *tool* to extract information hidden in the seismic data. It is in the latter stage of *knowledge extraction* where inversion takes place. It means inferring information about our model parameters from the seismic data.

This is a generic overview of the project, which involves the simulation of seismic waves propagating through media, also known as full waveform modelling. Seismic waves are elastic displacements or vibrations in the Earth. In an elastic material at least two types of waves can propagate: pressure waves and shear waves, whereas in an acoustic material only pressure waves do. Both types are governed by the same equation, the wave equation. It is peculiar to them the fact that their speed of propagation depends only upon the underlying media, and not upon their frequency or wavelength.

Nowadays, there are specific softwares which efficiently run such simulations in a wide range of scenarios. Namely, we will be using SPECFEM, which deploys the numerical spectral element method for full waveform modelling and allows the user to perform 2D and 3D simulations of acoustic, elastic, viscoelastic and poroelastic seismic wave propagation. In the project, we will design our own mesh structure with Gmsh (PyGmsh for Python interface) and run acoustic simulations only, which means set shear velocity to zero $v_s = 0$. From now on, every time we talk about velocity v, the pressure velocity v_p is implied.

Once we have the software, we are in a position to explore inversion methods that can be applied on the data we generate. Different scenarios can be posed. Through this project, we will work around the case of well-to-tie seismic data exploited and explained from the convolutional model. It consists in having a middle shot point surrounded by receiver

stations, which capture what happens around the middle trace. The signal recorded right at the shot point is called *zero-offset* trace, and accounts for the vertical phenomena only. With real borehole data, one usually has access to velocity and density models which can be translated into reflectivity, as we will see. The convolutional model will help us wrap everything as an inverse problem.

Chapter 2.

Inverse problems

The field of *inverse problems* spans those problems where data are used to compute the internal structure of physical objects. Conversely, the *forward problem* consists on predicting the outcome of some measurements given a complete description of a physical system.

While the forward problem is deterministic in the sense that its solution is unique, the inverse problem is not (Tarantola, 2005). Let us consider measurements of the gravity field around a planet, a recurrent example to illustrate inversion. According to Newton, given the distribution of the mass inside the planet, we can uniquely predict the values of the gravity (forward problem). However, there are different mass distributions that give that same value of the gravity (inverse problem). Therefore, the inverse problem has multiple solutions. This leads to the need of any prior information of the parameters (distribution of mass) that we may have.

The first step in any approach is to describe the problem through the forward problem. In its most general way, we could express it as:

$$\mathbf{d} = g(\mathbf{m}) \tag{2.1}$$

where $\mathbf{d}$ is the data, $\mathbf{m}$ the parameters and g the forward problem. As said, many (not to say infinite) sets of parameters $\mathbf{m}$ reproduce $\mathbf{d}$ within some uncertainty. The one that preforms the best will be the one that minimizes the *misfit*, which is the mismatch between the observed data and the result of a theoretical calculation:

$$E(\mathbf{m}) = \|\mathbf{d} - g(\mathbf{m})\|_2^2 \tag{2.2}$$

In order to minimize (2.2), various data fitting methods like least squares try to obtain the best match. However, one needs to be very careful in the representation of the data uncertainties, as Hadamard warned when he faced the heat conduction equation (Mosegaard, 2020). In his inversion approach, Hadamard found that small changes in the model parameters led to exploding changes in the predictions, causing the solution to be highly unstable. When this happens, we say that the inverse problem is *ill-posed*.

Ill-posed problems are the weak point of methods like least squares. Least squares are popular for solving inverse problems because they lead to the easiest computations. Their drawback is the lack of robustness, i.e, their strong sensitivity to a small number of large errors (outliers) in the data, or to the non-uniqueness of the solution.

Because of this, methods of regularization like *Tikhonov regularization* are required. They reformulate the ill-posed problem for numerical treatment, which typically involves including additional assumptions.

2.1 Tikhonov regularization

Tikhonov regularization is a typical regularization technique in which an extra expression is added to the least squares misfit function that controls the prior knowledge introduced through a regularization parameter.

We shall consider the linear case of expression (2.1), as it is the type of problem we will be dealing with during the whole project:

$$\mathbf{d} = \mathbf{G}\mathbf{m} \tag{2.3}$$

If the problem is ill-posed, the least squares estimation leads to an *overdetermined* and/or *underdetermined* system of equations. Let us discuss a bit more about this.

When there are more data than unknowns (parameters), there is not an exact solution and the problem is overdetermined. The matrix $\mathbf{G}$ maps the parameter space into a subspace of possible estimates. The least squares solution will find the solution that minimizes the distance between the true data and the estimated data $\|\mathbf{d} - \hat{\mathbf{d}}\|^2$, where $\hat{\mathbf{d}} = \mathbf{G}\hat{\mathbf{m}}$. It also means that it will try to fit the noise present in the data.

Under this situation, the linear least squares problem is to minimize:

$$E(\mathbf{m}) = \|\mathbf{d} - \mathbf{G}\mathbf{m}\|_2^2 = (\mathbf{d} - \mathbf{G}\mathbf{m})^{\mathrm{T}}(\mathbf{d} - \mathbf{G}\mathbf{m}) = \sum_{i=1}^{N}\left[d_i - \sum_{j=1}^{M} G_{ij}m_j\right]\left[d_i - \sum_{k=1}^{M} G_{ik}m_k\right] \tag{2.4}$$

whose solution $\hat{\mathbf{m}}$ can be found analytically by solving

$$\forall q : \frac{\partial E}{\partial \hat{\mathbf{m}}_q} = 0 \tag{2.5}$$

Following (Menke, 2012) one ends up solving the equation

$$\mathbf{G}^{\mathrm{T}}\mathbf{G}\mathbf{m} - \mathbf{G}^{\mathrm{T}}\mathbf{d} = 0 \tag{2.6}$$

Presuming that $\mathbf{G}^{\mathrm{T}}\mathbf{G}$ exists, we get the following estimate for the model parameters:

$$\hat{\mathbf{m}} = \left[\mathbf{G}^{\mathrm{T}}\mathbf{G}\right]^{-1}\mathbf{G}^{\mathrm{T}}\mathbf{d} \tag{2.7}$$

This is the least squares solution for an overdetermined problem, which, as said, is unstable towards ill-posed problems.

On the other hand, when there is too few data to determine uniquely all the model parameters, the problem is underdetermined. In this case there will be a whole subspace of

infinite points where all of them are equally good. In order to single out precisely one of the infinite number of solutions, we must add to the problem some external *a priori* information not contained in $\mathbf{Gm} = \mathbf{d}$. This is, information that is not based on the actual data.

For instance, consider the previous case of the gravity and the distribution of mass. Even without making any measurements, one can state with certainty that the density is everywhere positive. Furthermore, if we have a broad knowledge of the materials it is made of, we can establish some range known to characterize such materials.

The first kind of *a priori* assumption we shall consider is the expectation that the solution to the inverse problem is *simple*, where the notion of simplicity is quantified by some measure of the length of the solution (Menke, 2012). One such measure is simply the Euclidean length of the solution $L = \mathbf{m}^{\mathrm{T}}\mathbf{m} = \sum_i m_i^2$. So, a solution is therefore defined to be simple if it is small when measured under L_2 norm. Realistic or not, it can be useful occasionally, and we shall describe later on how it can be suitably generalized.

(Menke, 2012) poses the following minimization problem: find $\hat{\mathbf{m}}$ that minimizes $L = \mathbf{m}^{\mathrm{T}}\mathbf{m}$ subject to the constraint $\mathbf{E} = \mathbf{d} - \mathbf{Gm} = \mathbf{0}$. This can be solved using the method of Lagrange multipliers:

$$\Phi(\mathbf{m}) = L + \sum_{i=1}^{N} \lambda_i E_i = \sum_{i=1}^{M} m_i^2 + \sum_{i=1}^{N} \lambda_i \left[d_i - \sum_{j=1}^{M} G_{ij} m_j \right] \tag{2.8}$$

where λ_i are the Lagrangian multipliers. Minimizing with respect to m_q and taking the derivatives one obtains:

$$\frac{\partial \Phi}{\partial m_q} = \sum_{i=1}^{M} 2 \frac{\partial m_i}{\partial m_q} m_i - \sum_{i=1}^{N} \lambda_i \sum_{j=1}^{M} G_{ij} \frac{\partial m_j}{\partial m_q} = 2m_q - \sum_{i=1}^{N} \lambda_i G_{iq} \tag{2.9}$$

Setting this to zero and rewriting it in matrix notation yield the equation $2\mathbf{m} = \mathbf{G}^{\mathrm{T}}\lambda$, which must be solved along with the constraint equation $\mathbf{Gm} = \mathbf{d}$. Plugging the first equation into the second one gives $\mathbf{d} = \mathbf{Gm} = \mathbf{G}[\mathbf{G}^{\mathrm{T}}\lambda/2]$. If the inverse of the square matrix $\mathbf{GG}^{\mathrm{T}}$ exists, we can then solve this equation for the Lagrange multipliers, $\lambda = 2[\mathbf{GG}^{\mathrm{T}}]^{-1}\mathbf{d}$. Inserting this expression into the first equation yields the solution

$$\hat{\mathbf{m}} = \mathbf{G}^{\mathrm{T}} \left[\mathbf{GG}^{\mathrm{T}} \right]^{-1} \mathbf{d} \tag{2.10}$$

For a solution to exist, the problem $\mathbf{Gm} = \mathbf{d}$ must be purely underdetermined, which is that the corresponding system of equations contain no inconsistencies. However, this is not the case for most of the problems.

In general, most inverse problems are neither completely underdetermined nor completely overdetermined, i.e., *mixed-determined* problems. This is where Tikhonov regularization steps in. We would like to sort the unknown model parameters into two groups: those that are overdetermined and those that are underdetermined. In order to give preference to a

particular solution with desirable properties, we determine a solution that minimizes some combination Φ of the prediction error and the solution length for the model parameters:

$$\Phi(\mathbf{m}) = E - \varepsilon^2 L = \|\mathbf{d} - \mathbf{G}\mathbf{m}\|_2^2 + \varepsilon^2 \|\mathbf{m}\|_2^2 \tag{2.11}$$

where ε is the so-called *regularization parameter*, which controls the relative importance given to the prediction error and solution length.

Large values of ε will minimize the underdetermined part of the solution, but it also tends to minimize the overdetermined part of the solution. As a result, the solution will not minimize the prediction error E and will not be a very good estimate of the true model parameters. If ε is set to zero, the prediction error will be minimized, but no *a priori* information will be provided to single out the underdetermined model parameters. There is no simple method to determine a suitable value of ε, it must be done by trial and error.

Minimizing (2.11) in a way analogous to the least squares derivation (2.7), we obtain:

$$\hat{\mathbf{m}} = \left[\mathbf{G}^{\mathrm{T}}\mathbf{G} + \varepsilon^2 \mathbf{I}\right]^{-1} \mathbf{G}^{\mathrm{T}}\mathbf{d} \tag{2.12}$$

This is the Tikhonov regularization method, under the constraint of solutions with small L_2 norms. We have the tool, but the actual formulation of the problem is yet to be described. Our approach is based off the convolutional model, discussed in the next section.

Chapter 3.

Convolutional model

In this section we discuss the convolutional model as our approach to infer information from the data. It is a basic one-dimensional model that assumes that the seismic trace is simply the convolution of Earth's reflectivity $r(t)$ with a seismic source function $w(t)$ with the addition of a noise component $\eta(t)$. Mathematically, this can be expressed as:

$$s(t) = w(t) * r(t) + \eta(t) \tag{3.1}$$

We can also picture the case of having noisy reflectivity data. In such case the noise would be convolved with the wavelet too:

$$s(t) = w(t) * (r(t) + \eta(t)) = w(t) * r(t) + w(t) * \eta(t) \tag{3.2}$$

At first, we shall assume the noise component to be zero, in which case the seismic trace is simply the convolution of the wavelet with the Earth's reflectivity:

$$s(t) = w(t) * r(t) = \int_{-\infty}^{\infty} w(t-\tau) r(\tau)\, \mathrm{d}\tau \tag{3.3}$$

Figure 3.1 illustrates convolution (3.3) with actual data. From left to right, we see a reflectivity profile that is convolved with some wavelet, yielding the synthetic seismogram shown. Additionally, it should be noted that convolution is symmetric, which translates into $s(t) = w(t) * r(t) = r(t) * w(t)$.

Let us now discretize equation (3.3) by means of a quadrature sum on a uniform grid, and store the signal at equidistant points:

$$s_i = \sum_{j=0}^{n-1} w_j r_{i-j} \tag{3.4}$$

which is called discrete convolution. Discrete convolution can also be thought of as matrix-by-vector multiplication:

$$\mathbf{s}(t) = \mathbf{R}\boldsymbol{w}(t) \tag{3.5}$$

where $\mathbf{R}$ is known as Toeplitz matrix and contains the reflectivity coefficients arranged in the matrix such that the convolution operation is preserved. The elements of $\mathbf{R}$ are given as $R_{ij} = r_{i-j}$, i.e., they depend only on the difference between the row index

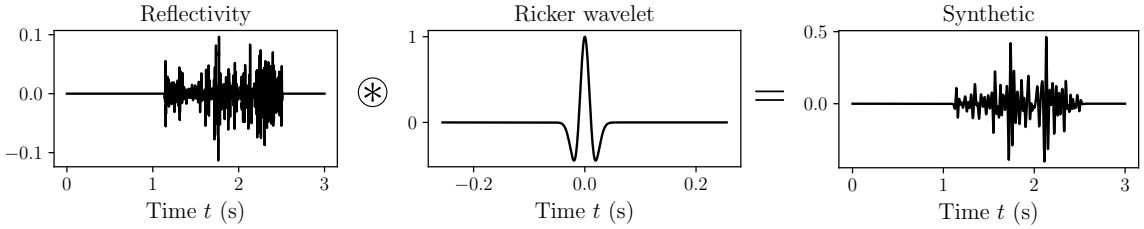


Figure 3.1.: Seismic trace in time domain as the convolution of the reflectivity with a time source function (wavelet).

and the column index. Assuming the reflectivity has zero entries in times prior to zero $r_{-1} = r_{-2} = \cdots = r_{1-n} = 0$, the Toeplitz matrix looks like:

$$\mathbf{R} = \begin{pmatrix} r_0 & 0 & 0 & \cdots & 0 \\ r_1 & r_0 & 0 & \cdots & 0 \\ r_2 & r_1 & r_0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ r_{n-1} & r_{n-2} & r_{n-3} & \cdots & r_0 \end{pmatrix} \tag{3.6}$$

If we define the Fourier Transform (*spectrum*) of a function $h(t)$ as:

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{-2\pi i f t} \mathrm{d}t \tag{3.7}$$

we obtain another but equivalent picture of $s(t)$ which takes place in the frequency domain. So, the Fourier Transform applied to equation (3.3) yield:

$$S(f) = W(f) \cdot R(f) \tag{3.8}$$

where $S(f)$, $W(f)$ and $R(f)$ are the Fourier transforms of $s(t)$, $w(t)$ and $r(t)$, respectively. Note that convolution becomes multiplication in the frequency domain. However, Fourier transform is a complex function, so we would have the amplitude and phase θ spectra:

$$\begin{aligned} |S(f)| &= |W(f)| \cdot |R(f)| \\ \theta_s(f) &= \theta_w(f) + \theta_r(f) \end{aligned} \tag{3.9}$$

3.1 Reflection coefficient series

The reflection coefficient series or reflectivity is one of the fundamental pillars in this seismic method. Basically, each reflection coefficient may be thought of as the response of the seismic wavelet to an acoustic impedance change within the Earth, where impedance is defined as the product of the compressional velocity and density. Mathematically, the mapping between acoustic impedance and reflectivity entails dividing the difference in

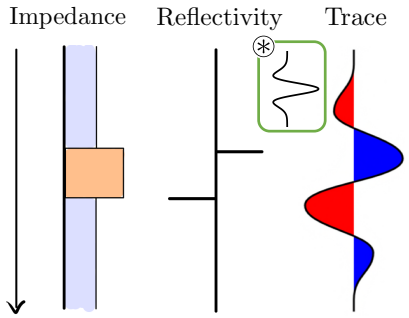


Figure 3.2.: Illustration of the reflectivity as delta functions when there is a change in the media. This is the ideal case, and as soon as there is noise present it differs from a delta function and broadens.

the acoustic impedances by the sum of the acoustic impedances. This gives the reflection coefficient at the boundary between two layers:

$$r_i = \frac{\rho_{i+1}v_{i+1} - \rho_i v_i}{\rho_{i+1}v_{i+1} + \rho_i v_i} = \frac{Z_{i+1} - Z_i}{Z_{i+1} + Z_i} \tag{3.10}$$

where ρ, v and Z are density, compressional velocity and acoustic impedance, respectively.

However, equation (3.10) is expressed in the depth domain, which means that all the model quantities depend upon the spatial variable only. We must convert from depth to time by integrating the sonic log travel times or slowness, which is the inverse of the velocity. Then we are in a position to apply equation (3.3). Time-to-depth conversion is a recurrent procedure during the calculations, and more details can be found in Appendix A.

Ideally, the reflectivity can be interpreted as a sum of delta functions: $r(t) = \sum_i r_i \delta(t - t_i)$, where r_i are the different reflection coefficients at each interface after time-depth conversion. It is later masked out through the wavelet, as Figure 3.2 illustrates.

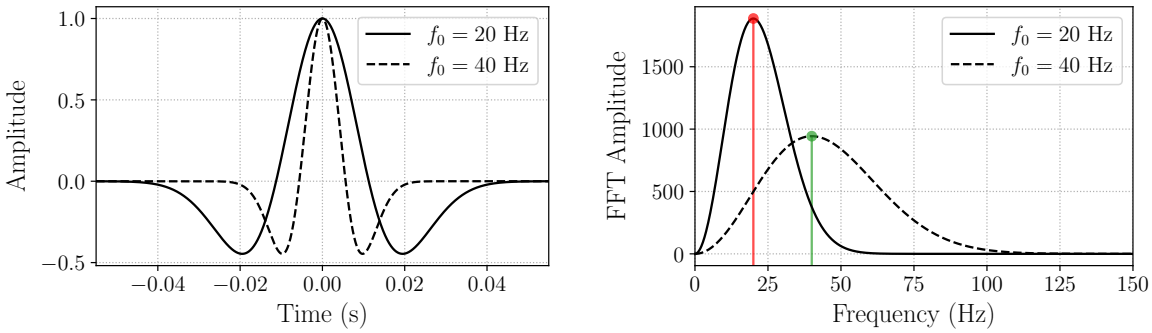
It is important to realize that equation (3.10) applies only to normal incidence cases. This means that only vertical phenomena is accounted for, which is a major shortcoming in the convolutional approach. This is an aspect that is discussed later.

Once we have two of the three quantities in equation (3.3) at our disposal, such as the wavelet and the recorded seismogram, we can infer the third one through inversion. Mathematically speaking, it does not make much difference whether to infer the wavelet or the reflectivity since convolution is symmetric.

3.2 Seismic wavelet

There is not a single, well-defined wavelet which produces the seismic trace. The wavelet is, overall, both time-varying and complex in shape. However, the assumption of a simple wavelet at first is reasonable. In our case, the Ricker wavelet will be our choice of simple wavelet.

The Ricker wavelet, also referred to as Mexican hat, consists of a peak and two side lobes as shown in Figure 3.3a. The Ricker wavelet is dependent only on its dominant frequency, that is the peak frequency of its amplitude spectrum or the inverse of the dominant period



(a) Ricker wavelet.

(b) Spectrum of Ricker wavelet.

Figure 3.3.: Ricker wavelet. (a) In time domain for two different dominant frequencies 20 Hz and 40 Hz. (b) In frequency domain, i.e., Fourier space. We see that they peak at the dominant frequency in each case: 20 Hz (red) and 40 Hz (green).

in the time domain. The dominant frequency provides the resolution power: the smaller it is, the more events will capture when convolved with the reflectivity. Figures 3.3b shows exactly this in the frequency domain.

Mathematically, the Ricker wavelet is the second derivative of the Gaussian function $e^{-t^2/2}$, which can be expressed as:

$$\psi(t) = \left(1 - 2at^2\right) e^{-at^2} \text{ with } a = (\pi f_0)^2 \tag{3.11}$$

where f_0 is the dominant frequency.

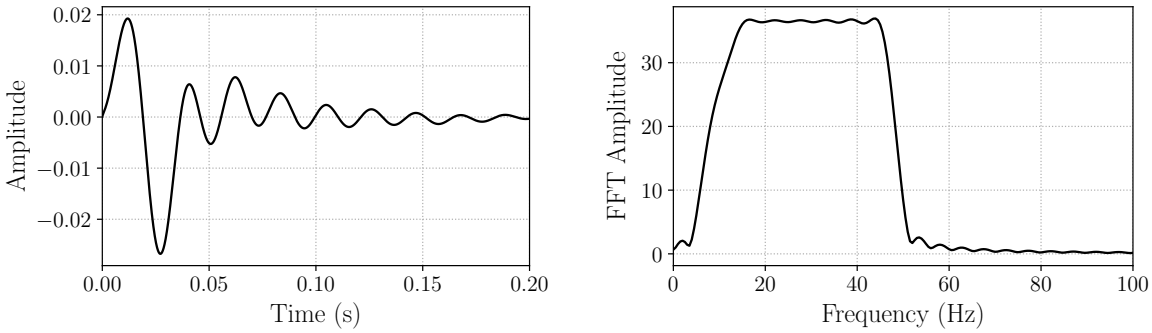
The Ricker wavelets are zero-phase, or perfectly symmetrical. This is a desirable characteristic of wavelets since the energy is then concentrated at a positive peak, and the convolution of the wavelet with a reflection coefficient will better resolve that reflection. Zero-phase wavelets have energy before time zero, which makes them non-causal, and hence not physically realizable.

On the other hand, minimum-phase wavelets do have physical meaning and, therefore, are the ones found in real life. Unlike the zero-phase kind, minimum-phase wavelets are causal and their energy is zero before time zero. This kind of wavelets are of great importance, since if the source wavelet is not minimum-phase, the results are likely to be unstable as we will discuss. Figure 3.4 shows an example of minimum-phase wavelet.

3.3 Noise component

The situation that has been discussed so far is the ideal noiseless case. That is, we have interpreted every reflection on a seismic trace as being an actual reflection from a lithological boundary. Actually, many of the "wiggles" on a trace are not true reflections, but are actually the result of seismic noise. Seismic noise can be grouped under two categories:

- **Random noise**. Noise which is uncorrelated from trace to trace and is due to mainly environmental factors.



(a) Minimum-phase time source. **(b)** Spectrum.

Figure 3.4.: Minimum-phase wavelet. We can see on the left panel (a), the energy is zero prior to time zero, rendering it causal and physically realizable.

- **Coherent noise**. Noise which is predictable on the seismic trace but is unwanted. An example is multiple reflection interference.

Random noise can be thought of as the additive component $\eta(t)$ in equation (3.1). Correcting for this term is the primary reason for stacking our data. As we will see, stacking actually does an excellent job at removing random noise.

Multiples, one of the major sources of coherent noise, are caused by multiple "bounces" of the seismic signal within the Earth. Multiples cannot be thought of as additive noise and must be modelled as a convolution with the reflectivity.

3.4 Deconvolution as an inverse problem

So far, the 1D convolutional model has been introduced, which is mathematically expressed through equation (3.1). It is overly simple, but also a convenient starting point for inversion. Thus, it is possible to showcase its early difficulties and limitations.

We shall start off with the ideal noiseless case from equation (3.3). As said before, seismograms serve as a useful *seismic image*, but wavelets are filters that keep us from seeing what we set out to see in seismic data: the Earth's reflectivity. Through deconvolution, we try to remove the wavelet from the data. Any noise that might be present further complicates the problem.

When well logs are known, we should be able to compute an impedance profile which corresponds to the seismic traces near the well control. The impedance log can then be used to estimate the reflectivity sequence. Under the assumption of horizontal layers, equation (3.10) can be used for this task. Once the reflectivity sequence is known, the wavelet may be estimated using a number of methods.

One first approach is spectral division. It involves equation (3.8) to find $W(f)$ in the frequency domain:

$$W(f) = \frac{S(f)}{R(f)} \tag{3.12}$$

However, it is usually the case that $R(f)$ is zero for some values of f. In this case, $W(f)$ is approximately given by:

$$W(f) = \frac{S(f)R^*(f)}{R(f)R^*(f) + \varepsilon^2} = \frac{S(f)R^*(f)}{|R(f)|^2 + \varepsilon^2} \tag{3.13}$$

where $W^*(f)$ is the complex conjugate of $W(f)$ and ε^2 a small positive constant chosen large enough to secure a numerically stable solution (Mosegaard, 2012). The wavelet $w(t)$ can now be computed via the inverse Fourier Transform:

$$w(t) = \int_{-\infty}^{\infty} W(f) e^{2\pi i f t} \mathrm{d}f \tag{3.14}$$

Alternatively, we can solve the problem in the time domain as a least squares problem. Equation (3.5) represents the linear forward problem of convolution, which can be identified with (2.3). Here, $\mathbf{s}$ is the seismic data and $\boldsymbol{w}$ the set of parameters to estimate, linked together through $\mathbf{R}$. This allows us to solve the problem deterministically through Tikhonov regularization (2.12):

$$\hat{\boldsymbol{w}} = \left[\mathbf{R}^{\mathrm{T}}\mathbf{R} + \varepsilon^2\mathbf{I}\right]^{-1} \mathbf{R}^{\mathrm{T}}\mathbf{s} \tag{3.15}$$

where $\hat{\boldsymbol{w}}$ is the estimated wavelet. The inversion scheme may be set up by forward modeling to compute the seismogram. The synthetic seismogram is then compared to the seismic trace, and we can assess how close we are to the solution.

Deterministic wavelet methods must make a number of estimates which leave some questions regarding the final wave shape left on the data. Equation (2.12) was derived under the assumption that the solution must be simple, which is the reason why the regularization term $\|\mathbf{m}\|_2^2$ comes about in equation (2.11). However, it could be that simplicity is not enough for regularization to be able to find the right solution. As a mixed-determined problem, external (prior) information must be added according to the problem. In our case, some *a priori* information could be compact support of the wavelet, which means sufficiently fast decay to obtain localization (Foufoula-Georgiou and Kumar, 1994). Such measure of compact support can be introduced by redefining L in equation (2.11). The compact support of $\boldsymbol{w}$ can be ensured by applying exponentially growing weights $ar^{k\cdot i}$ to its components $\{w_0, w_1, \ldots, w_i, \ldots, w_M\}$, so that the tails or long term high energy terms are heavily penalized:

$$\boldsymbol{w}_c = a \begin{pmatrix} r^{k\cdot 0} & 0 & \cdots & 0 & \cdots & 0 \\ 0 & r^{k\cdot 1} & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & r^{k\cdot i} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \cdots & r^{k\cdot M} \end{pmatrix} \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_i \\ \vdots \\ w_M \end{pmatrix} = \mathbf{C}\boldsymbol{w} \tag{3.16}$$

where the matrix $\mathbf{C}$ contains the measure of compact support. The overall compact support of the solution is then:

$$L_{\text{weighted}} = \boldsymbol{w}_c^{\mathrm{T}} \boldsymbol{w}_c = [\mathbf{C}\mathbf{w}]^{\mathrm{T}} [\mathbf{C}\mathbf{w}] = \boldsymbol{w}^{\mathrm{T}} \mathbf{C}^{\mathrm{T}} \mathbf{C} \boldsymbol{w} = \boldsymbol{w}^{\mathrm{T}} \mathbf{W} \boldsymbol{w} \doteq \|\boldsymbol{w}\|^2_{\text{weighted}} \tag{3.17}$$

The matrix $\mathbf{W} = \mathbf{C}^{\mathrm{T}}\mathbf{C}$ can be interpreted as a weighting factor that comes about in the calculation of the $\boldsymbol{w}$-norm $\|\boldsymbol{w}\|^2_{\text{weighted}}$. We just need to find the right values of the parameters a, r and k that enter into the diagonal of $\mathbf{C}$ in equation (3.16).

The inverse problem solutions derived in (2.10) and (2.12) can be modified to take into account *a priori* information of the solution. The derivations are substantially the same as for the unweighted case (Menke, 2012, pp. 56–58), which results in:

- Weighted undetermined solution:

$$\hat{\boldsymbol{w}} = \mathbf{W}^{-1}\mathbf{R}^{\mathrm{T}} \left[\mathbf{R}\mathbf{W}^{-1}\mathbf{R}^{\mathrm{T}}\right]^{-1} \mathbf{s} \tag{3.18}$$

- Weighted Tikhonov solution. It is solved by minimizing $\Phi(\boldsymbol{w}) = E - \varepsilon^2 L_{\text{weighted}}$, where $L_{\text{weighted}} = \|\boldsymbol{w}\|^2_{\text{weighted}} = \boldsymbol{w}^{\mathrm{T}}\mathbf{W}\boldsymbol{w}$:

$$\hat{\boldsymbol{w}} = \left[\mathbf{R}^{\mathrm{T}}\mathbf{R} + \varepsilon^2\mathbf{W}\right]^{-1} \mathbf{R}^{\mathrm{T}}\mathbf{s} \tag{3.19}$$

Chapter 4.

Inversion of real 1D data

The wavelet is a key factor in logging-constrained seismic inversion. Our approach for seismic wavelet extraction is to use the least squares method to calculate the wavelet based on the existing logging data and adjacent seismic records. This approach is affected by seismic noise and logging errors, so we will exploit the addition of noise to the model.

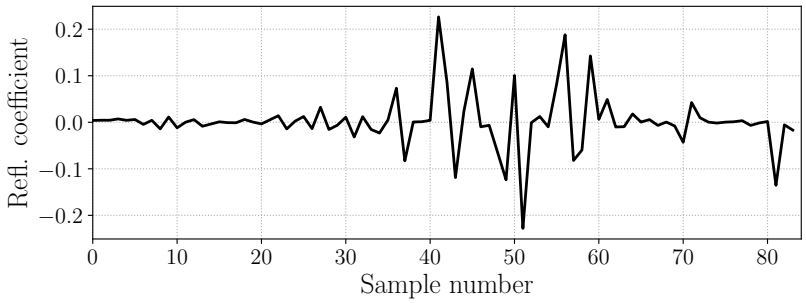
We shall start off by deploying some simple, but real, one-dimensional data as a first approach. We have at our disposal the reflectivity (from well 'Løgumkloster-1' in South Jutland) and the wavelet (from DRUM dataset in Scotland). They are shown in Figure 4.1 and are "free of noise". In quotations because some noise might be inherent, but we do not know it and, if so, we do not know its origin. Mention that the wavelet array 4.1b has been re-sampled so that it has got the same time duration as the reflectivity 4.1a.

Let us now address the inverse problem $\mathbf{s} = \mathbf{R}\boldsymbol{w}$, aimed at finding $\hat{\boldsymbol{w}}$ from the data. Since the true wavelet $\boldsymbol{w}$ is known, the inversion procedure can be set out so that the true seismic data $\mathbf{s}$ is synthetically computed through convolution in the first place (Figure 4.1c). Next is to use $\mathbf{s}$ to estimate the wavelet $\hat{\boldsymbol{w}}$ by any of the methods discussed in chapters 2 and 3. Lastly, the estimated data $\hat{\mathbf{s}} = \mathbf{R}\hat{\boldsymbol{w}}$ can be compared with the true, synthetic seismic trace.

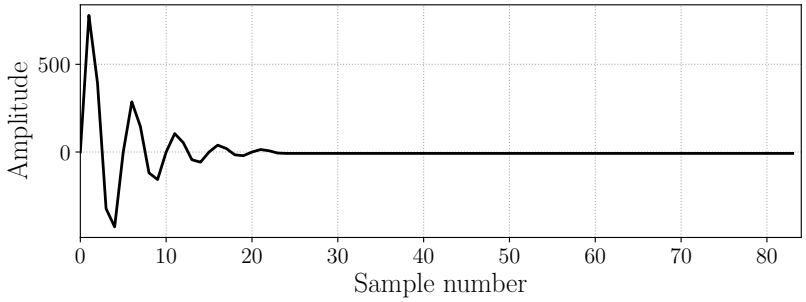
One of the methods is simply least squares, with no constraints and which only tries to fit the data. Often it is also referred to as the *naïve solution*. It means solving equation (2.7), which comes from minimizing $E(\boldsymbol{w}) = \|\mathbf{R}\hat{\boldsymbol{w}} - \mathbf{s}\|_2^2$. The naïve solution is exposed in Figure 4.2a in black. At first sight, we see that it already performs well as it fits *perfectly* the true wavelet (dotted blue line). The right panel of Figure 4.2a shows the distribution of the residuals, defined as $r_i = \hat{w}_i - w_i$. Thus, the mean-squared error can be defined as $\sigma_w^2 = M^{-1}\sum_{i=1}^{M} r_i^2 \simeq 47.5$, where M is the dimension of the wavelet array. This leads to an error in the wavelet estimate of $\sigma_w \simeq 6.9$.

Figure 4.2a also exposes the spectral division approach in red. Its solution is worse, as it shows non-zero energy terms in the long term. This makes the estimate not very useful, something that is borne out by the distribution of the residuals on the right. The deviation of the predictions increases fourfold with respect to the naïve solution: $\sigma_w \simeq 27.3$.

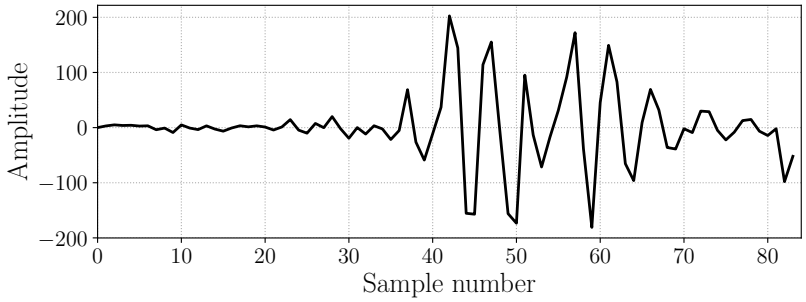
Let us see how well the estimates reproduce the data in Figure 4.2b. The naïve solution reproduces the data perfectly (red line): $\sigma_s \simeq 0.0$. This is the reason why its distribution of residuals is not plotted. In fact, the correlation coefficient between the real data and estimated data is 1.0 (so, full overlap). On the other hand, the spectral division (green line) seems to perform fairly well too, in spite of the long term reverberations. It gives a spread of about $\sigma_s \simeq 5.3$.



(a) Reflectivity profile.



(b) True wavelet.



(c) Seismogram after convolving the reflectivity (a) with the wavelet (b).

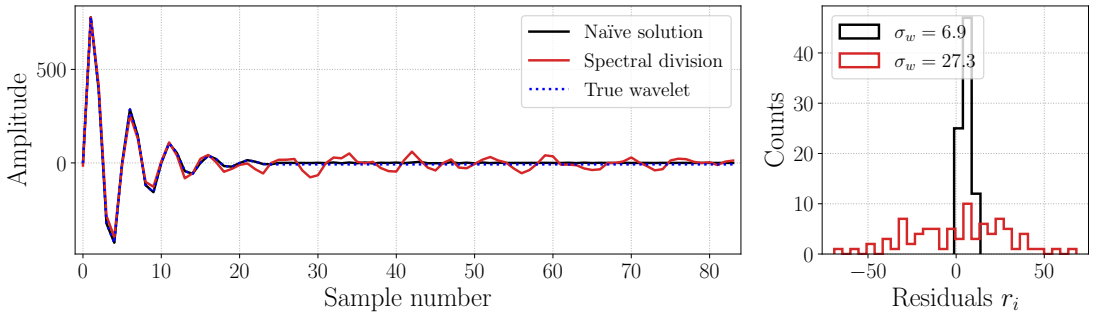
Figure 4.1.: Noise-free real data. We have access only to the reflectivity (a) and the wavelet (b). Then, the seismic trace (c) is obtained through convolution of these two.

So far, no regularization has been needed to find sensible solutions. Nonetheless, it was mentioned before that the least squares approach is not robust against noise. We will see that it tries to fit the noise present in the data by all means, leading to meaningless solutions. This is where regularization comes into play to render the process stable.

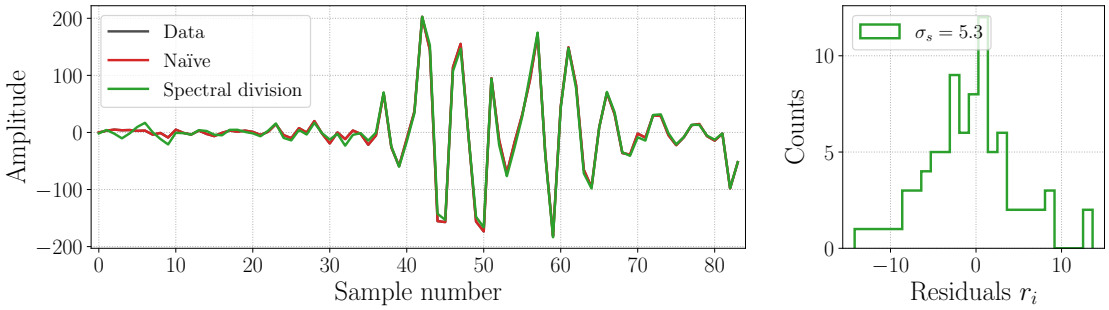
Let us introduce white Gaussian noise in the synthetic data, leaving the reflectivity intact. Thus, the noisy seismic data is

$$\mathbf{s}_{\mathcal{N}} = \mathbf{s} + c\|\mathbf{s}\|_2 \cdot \boldsymbol{\mathcal{N}}(\mu, \sigma) \tag{4.1}$$

where $\boldsymbol{\mathcal{N}}(\mu, \sigma)$ is the normalized Gaussian distribution of the noise and c a parameter that controls how much of it is introduced. The subsequent results in this section are obtained using the parameter values: $\mu = 0$, $\sigma = 0.5$ and $c = 0.2$.



(a) Wavelet estimates $\hat{\boldsymbol{w}}$.



(b) Data estimates $\hat{\mathbf{s}}$.

Figure 4.2.: Noise-free solutions. (a) On the left, wavelet estimate using the reflectivity 4.1a and the data 4.1c. The naïve solution in black, spectral division in red and true wavelet in dotted blue. On the right, the distribution of the residuals $r_i = \hat{w}_i - w_i$. (b) On the left, the predictions on the data. On the right, the distribution of residuals $r_i = \hat{s}_i - s_i$. Only the residuals of the spectral division are plotted.

The naïve solution with noisy data is plotted in Figure 4.3. As expected, the solution becomes very unstable as soon as we introduce noise. Least squares gets lost in the noise, as it tries to fit the data to its best (noise included). Thus, we do not get a meaningful wavelet because it strongly reverberates throughout its whole duration. The energy of the estimated wavelet $\|\hat{\boldsymbol{w}}\|_2 \simeq 1992$ almost doubles the energy of the true wavelet $\|\boldsymbol{w}\|_2 \simeq 1102$.

It is time to implement Tikhonov regularization, intended to add a constraint on the shape of the solution through the regularization term. First off, we shall inspect equation (3.15). It yields the left profile of Figure 4.4a, which is closer to be a wavelet since the dominant high energy terms are found at the beginning. A value of $\varepsilon = 0.05$ is chosen by *trial and error* until we get a reasonable solution. Even though ε is meant to reduce those long term reverberations, there are still undesirable non-zero values along the tail. The reason is that for this solution only the simplicity constraint of the wavelet applies ($L = \mathbf{m}^{\mathrm{T}}\mathbf{m}$), which seems to not be enough to find a good solution in the parameter space. The right panel of Figure 4.4a shows the distribution of the residuals. Although plain Tikhonov outperforms the naïve solution, there is room for improvement.

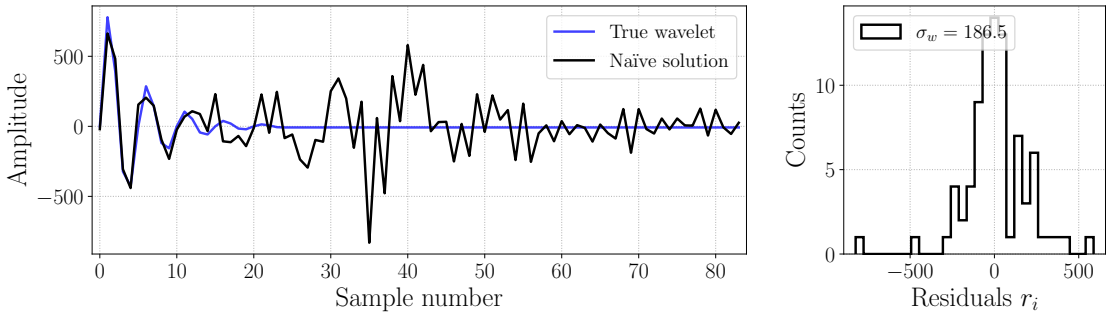


Figure 4.3.: Naïve wavelet estimate with noisy data.

The wavelet estimate from Figure 4.4a results in the red estimate of the data in Figure 4.4c. Overall, it reproduces the data fairly well, something that is backed up by the distribution of the residual. Yet, we still appreciate some noise overfitting.

Ideally, we would like the solution to meet compact support, and have zero energy terms in the tail. Provided we know the length of the wavelet, one quick, but *cheating*, way of achieving this is by forcing all the wavelet energy to be in the first few samples by means of a projector operator. The projector operator $\hat{\boldsymbol{\mathcal{P}}}$ is an $N \times N$ matrix with all of its entries zero but the first elements on the diagonal corresponding to the first few samples where we want the energy to be concentrated. These positions are filled with ones, so we are basically killing the terms with zeros on the diagonal. Then, we apply it to $\mathbf{R}$ to obtain the projection:

$$\mathbf{R}_{\mathcal{P}} = \hat{\boldsymbol{\mathcal{P}}} \cdot \mathbf{R} \tag{4.2}$$

Now, we only have to replace $\mathbf{R}$ with $\mathbf{R}_{\mathcal{P}}$ in equation (3.15). The resulting wavelet estimate is exposed in Figure 4.4b. As expected, all the energy is located in the first few samples, giving it the wavelet-like shape we aim for. So, thus Tikhonov is able to get a closer match with a smaller deviation σ_w. We force it to happen though. Not surprisingly, in Figure 4.4c we see in green a slightly better estimate of the data that gives $\sigma \simeq 10.0$.

However, at the end of the day, this is not a good practice as we are not introducing prior information. We are rather interested in introducing *a priori* information that penalizes some solutions over others, helping better Tikhonov find the right solution.

Weighted Tikhonov regularization

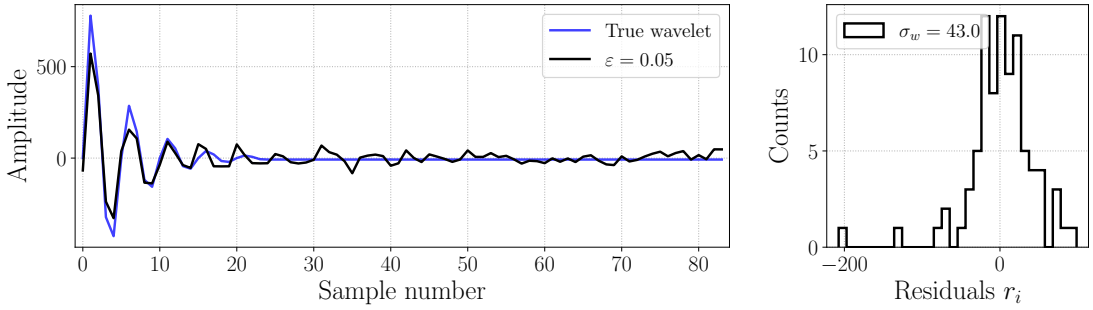
In section 3.4 the $\mathbf{W}$ matrix came in as a way of introducing prior information deterministically. This results in solving equation (3.19), which, in a nutshell, minimizes the weighted length $L = \boldsymbol{w}^{\mathrm{T}}\mathbf{W}\boldsymbol{w} = \|\boldsymbol{w}\|^2_{\text{weighted}}$ under the misfit constraint $E(\boldsymbol{w}) = \|\mathbf{s} - \mathbf{R}\boldsymbol{w}\|_2^2$.

The reasoning is the same as before, but some fine-tuning of ε and $\{a, r, k\}$ is required in order to find the best solution, where $\{a, r, k\}$ are the exponential parameters that enter into equation (3.16). For the shake of simplicity, $a = 1$ and $r = e$ are fixed, so that the weights on the diagonal are ruled by $e^{k \cdot i}$.

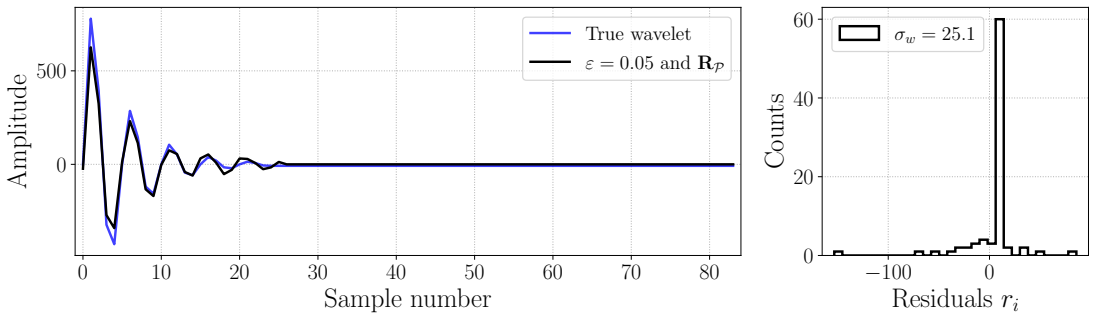
Let us now inspect a range of values for both ε and k to get kind of a *landscape* of the misfit. Then, we only have to get the minimum of the landscape to obtain the optimal values of $\{\varepsilon, k\}$. However, by doing only this we fall back into the problem of overfitting the noise. We would like to fit the data within some uncertainty Δ: $\|\mathbf{s} - \mathbf{R}\hat{\boldsymbol{w}}\|_2^2 \simeq N\Delta^2$. Hence, we have to find the minimum of the misfit landscape that allows for Δ, i.e., the minimum of the quantity $\|\mathbf{s} - \mathbf{R}\hat{\boldsymbol{w}}\|_2^2 - N\Delta^2$, for a given value of Δ.

Figure 4.5 exhibits the landscape of the misfit. The ranges for ε and k have been chosen within the intervals $[0.01, 2.5]$ and $[0.01, 5]$, respectively. The red dot is where minimum of the misfit lies: $(\varepsilon_0, k_0) = (0.28, 1.24)$. The minimum of the misfit has been found within an uncertainty of $\Delta = 10$. Additionally, it is reasonable that the misfit increases as we increase either ε or k, since we would be killing more and more desirable high energy terms at the beginning of the wavelet.

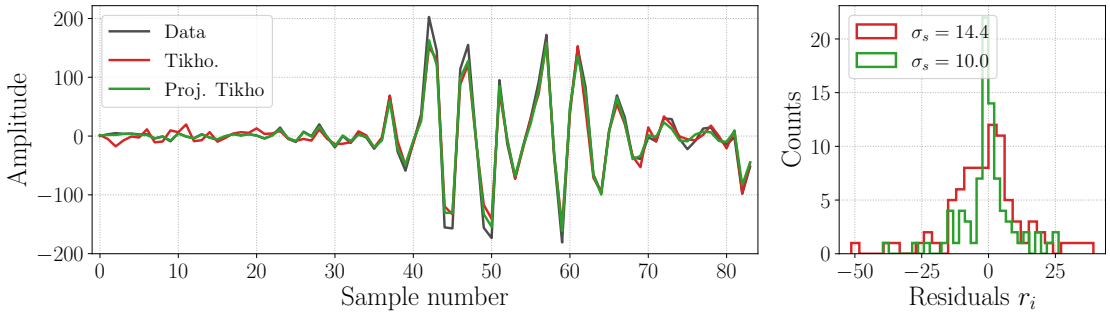
Figure 4.6 collects the results we get out of plugging in (ε_0, k_0). As expected, we get a solution that is stable against noise, and, therefore, it does not reverberate. All the energy is concentrated in the first few samples (Figure 4.6a), without the need of a projector operator. Consequently, it reproduces the data within the uncertainty we set in the first place: $\sigma \simeq \Delta \simeq 10.0$ (Figure 4.6b). For instance, by looking at the first 30 samples, the noise in the data (gray) manifests with many ups and downs which are not fitted by the estimated data.



(a) Wavelet estimate using Tikhonov.



(b) Wavelet estimate through $\mathbf{R}_\mathcal{P}$.



(c) Reproduced seismic data.

Figure 4.4.: Noisy Tikhonov solutions. (a) On the left panel, wavelet estimate using Tikhonov with regularization parameter $\varepsilon = 0.05$. On the right, the distribution of the residuals. (b) On the left, wavelet Tikhonov estimate using the projected reflectivity $\mathbf{R}_\mathcal{P}$ and $\varepsilon = 0.05$. On the right, distribution of the residuals. (c) Reproduced seismic data using the two wavelet estimates (a)-red and (b)-green. In gray, the *true* data. On the right panel, distribution of the residuals for both solutions.

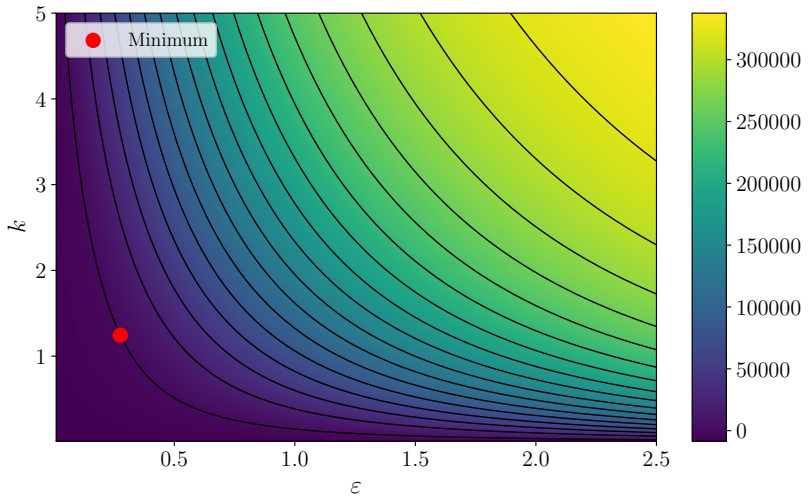
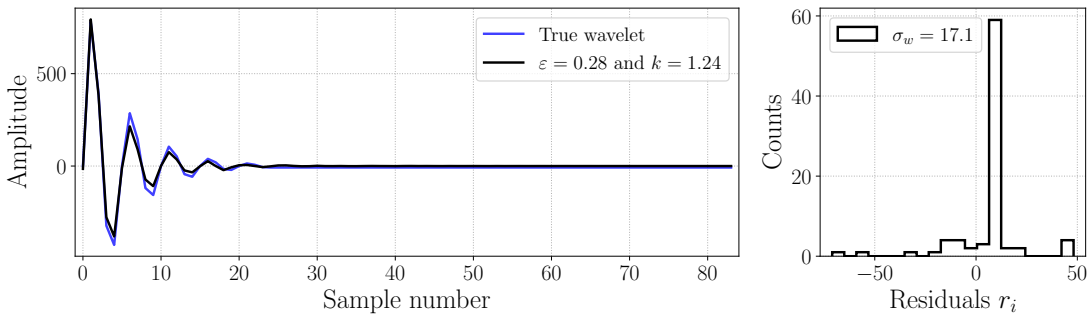
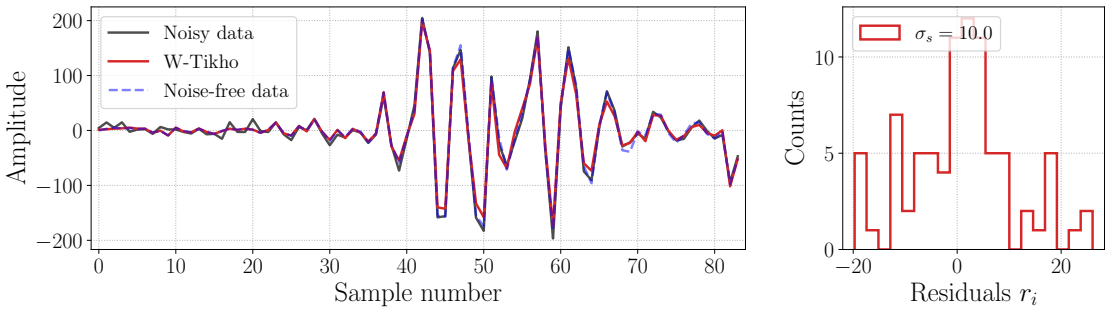


Figure 4.5.: Misfit landscape. The red dot marks the values of $\{\varepsilon, k\}$ which minimize $\|\mathbf{s} - \mathbf{R}\hat{\boldsymbol{w}}\|_2^2 - N\Delta^2$. The black lines are the contour lines of the surface.



(a) Wavelet estimate $\hat{\boldsymbol{w}}$.



(b) Data estimate $\hat{\mathbf{s}}$.

Figure 4.6.: Noisy weighted Tikhonov solutions. Parameter values are $\varepsilon = 0.28$ and $k = 1.24$. (a) Wavelet estimate, with the distribution of the residuals on the right. (b) Data estimate, with the distribution of the residuals on the right.

Chapter 5.

Simulation of 2D data

In section 4 we apply deterministic inversion tools on 1D real data. Yet, we know nothing about the structure of the Earth where the data was recorded, nor the velocity and density models. We can make assumptions based off the larger spikes of the reflection coefficients, meaning that there is some activity, and, consequently, the seismic waves are encountering some set of interfaces. Nevertheless, we do know that the previous inversion methods based on convolution work, so we might inquire up to what extent they perform well.

One way of testing models and results is by means of simulations. They allow us to come up with different structures of the Earth, run models on them and gather the seismic responses. These simulated seismic data are 2D and are deemed to be *real*, as they are meant to reproduce real life experiments. At the end of the day, simulations solve, in a very efficient manner, the wave equation on grids and structures that shape real life situations. We shall consider rather simple scenarios at first, and then build on the outcomes we get.

Learn how to use the actual software can be cumbersome, but that is a whole different story. In section 5.1 we mention briefly which software we will be using, as well as how to give shape to the whole problem itself: from the idea to the results.

Before going any further, we should bear in mind the early limitations of the convolutional model. It only accounts for linear phenomena that occurs vertically (normal wave reflections), so it does not explain nonlinear migration or any other geophysical effects that imply *horizontal communication*. So, we should correct for them first thing. This is why preprocessing is a crucial step.

The procedure is straightforward: we simulate data and preprocess it in such a way that in the end we are able to compare the simulated data with the convolutional data. If they agree up to some extent, then we should be in a position to apply inversion as described in section 3.4.

Final mention to the wavelet. Throughout the entire project the Ricker wavelet with a dominant frequency of 7 Hz will be employed. Higher frequency wavelets might be desired for higher resolution, but, for some reason, they would render the simulation unstable and extremely sensitive to the precision of the structure.

5.1 How do we simulate data?

We will leverage SPECFEM2D to run the 2D simulations. It also has its 3D counterpart (SPECFEM3D) for 3D modelling. It is a powerful tool to perform all sorts of simulations of

wave propagation, as well as full waveform imaging and adjoint tomography, but we are only interested in the part of simulating the data. Further information can be found in the API documentation (Komatitsch, 2018). Although it is written in Fortran, it contains a Python-like master script called *Parfile* which contains all the parameters and conditions that apply to our problem.

A mesh that keeps track of the structure we want to realize must be specified, but SPECFEM turns out to be a bit restrictive when it comes to using its inner meshing tool. Because of this, it also works in concert with external tools to import external grids. In our case we will use PyGmsh, which is the Python interface of Gmsh, a finite element mesh generator (Geuzaine and Remacle, 2002).

Parallelly, the velocity v and density ρ models must be created accordingly. They are defined in the depth domain, which, together with the grid, serve as an input for SPECFEM. Depth domain means that all the quantities, such as the velocity at each point or the positions of the layers, potentially depend on both spatial coordinates x and z. However, we get as an output the seismic data in the time domain. Moreover, the convolution equation (3.3) is also in the time domain. Hence, time-depth conversion is necessary. We must convert $v(x, z)$ and $\rho(x, z)$ into the time domain: $v(x_0, t)$ and $\rho(x_0, t)$. In the latter representation, the x_0 variable stands for the position of the receivers. A time-depth relationship can be constructed as:

$$t_i = t_{i-1} + \frac{\mathrm{d}z}{v_{i-1}} \quad \text{with} \quad z_i = z_{i-1} + \mathrm{d}z \tag{5.1}$$

This is, each time step t_i can be computed as the previous time step t_{i-1} plus the time it takes to go through the depth element $\mathrm{d}z$ (assuming it is the same everywhere) at velocity v_{i-1}. More detailed information on how to implement it can be found in the Appendix A.

Within the *Parfile* we can also specify the kind of boundary conditions. In our case, we will use absorbing boundary conditions. More technical details can be found at (Komatitsch, 2018, Section 3).

Finally, SPECFEM expects two more files: *SOURCE* and *STATIONS*. Inside *SOURCE*, we specify the coordinates of the shot point, as well as the time source or wavelet. Given that the final goal is to make a comparison with the convolutional model, the same wavelet should be used (spoiler: Ricker wavelet). On the other hand, the *STATIONS* file contains the coordinates of the receivers, so we will get as many gathers as the number of receivers. Note that if we place the source in the middle at the surface of the section, we will obtain CMP (common midpoint) gathers. CMP gathers are useful for our purposes, since the middle trace is *common* for all the receivers it will be very representative of what the rest of stations record.

Once all of the above is specified, we only have to compile and run. As an output, we can also specify the format of the seismic data, e.g., SU format. The time and precision of the simulation is governed by the number of time steps (N_t) and the duration of each time step ($\mathrm{d}t$), so that the one-way travel time is just $N_t \times \mathrm{d}t$.

5.2 Velocity and density models

In the previous section, it was stated that the v-ρ models in the depth domain must be specified. We would like to design a model that reproduces the reflection coefficients from Figure 4.1a. According to equation (3.10), we can set out a relationship between the reflection coefficients and the density and velocity values:

$$r_i = \frac{v_{i+1}\rho_{i+1} - v_i\rho_i}{v_{i+1}\rho_{i+1} + v_i\rho_i} \quad \longleftrightarrow \quad v_{i+1}\rho_{i+1} = \frac{r_i + 1}{r_i - 1} v_i\rho_i \tag{5.2}$$

Assuming some kind of relationship between the velocity and the density $v_i = \alpha\rho_i$, we get:

$$\rho_{i+1} = \sqrt{-\frac{r_i + 1}{r_i - 1}\frac{v_i\rho_i}{\alpha}} \tag{5.3}$$

By fixing α and setting some initial values (v_0, ρ_0), the rest of the series (v_i, ρ_i) would be completely characterized, and they would reproduce the desired reflection coefficients.

A simple geological model, in harmony with all of the above and which links to the simulations, is one that consists of horizontal layers. That is, as many horizontal interfaces as reflection coefficients. Through equation (5.3) we can attribute constant v-ρ values to each layer between two interfaces. This simple setting is 1D in the sense that it only depends on z. The resultant reflectivity looks like the one shown in Figure 5.1.

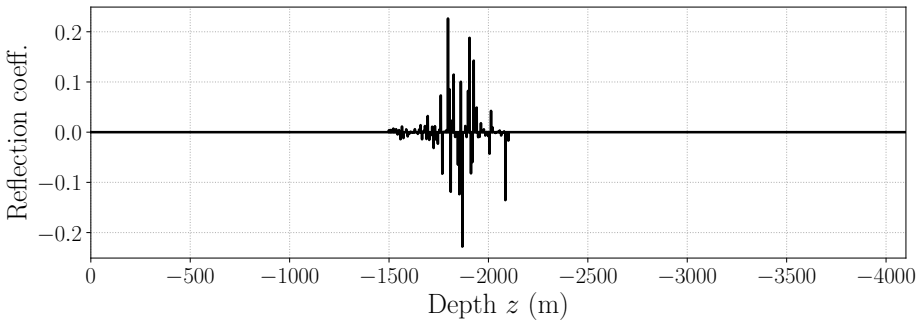
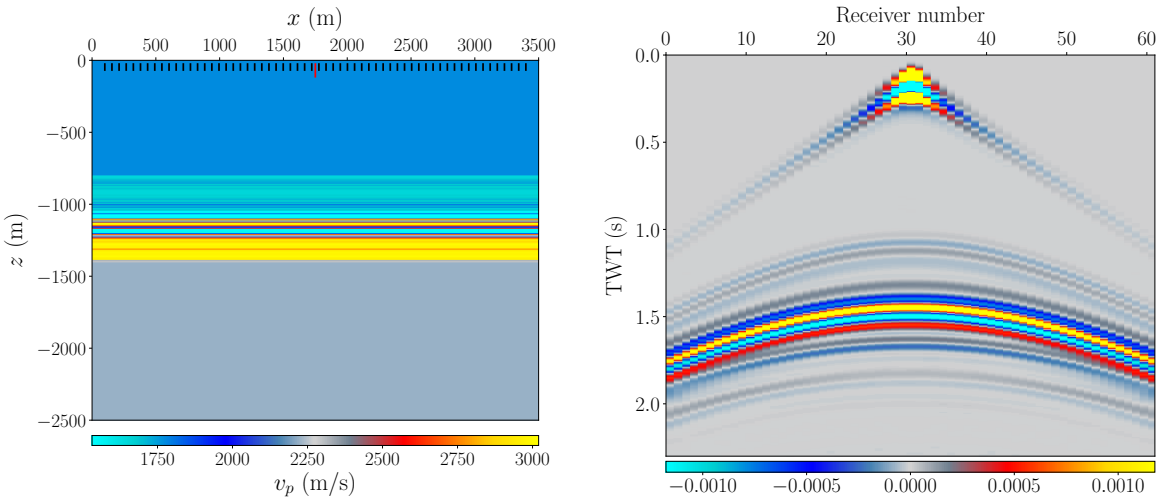


Figure 5.1.: Reflection coefficient series for the multilayer model.

Note that all the materials are acoustic, so it is implicit the foregoing velocities refer to the P velocity (v_p).

5.3 The need of preprocessing

Let us illustrate an example of a simulation. Figure 5.2 exposes the results using a horizontal multilayer structure. On the left, we see the velocity model in the depth domain. It has been conceived using (5.2) and (5.3), with $\alpha = 1.8$ and initial values $v_0 = 1750\,\mathrm{m/s}$ and $\rho_0 = 1000\,\mathrm{kg/m^3}$. The little stickers mark the position of the receivers. There are a total of 61 evenly spaced stations, positioned around the shot point in the middle (red stick). This structure leads to the seismic data shown on the right. The first thing we may notice is the uppermost signal which pertains to the direct wave travelling through the first layer.



(a) Velocity v_p model.

(b) Simulated data.

Figure 5.2.: Results from running the simulation on the horizontal multilayer model. (a) Velocity model in the depth domain. The source is located in the middle at the surface. The little sticks along the top point out the position of the stations. There are 50 stations evenly distributed around the shot point. (b) Simulated seismic data. TWT stands for two-way-travel time, which is the time it takes to reach the reflector and come back.

It does not provide any valuable information about the structure, so we only have to *mute* it, i.e., setting its amplitude to zero. This makes up the very first step in the preprocessing, also called *muting*.

Another noteworthy effect is the curvature that the data suffers, even though we are dealing with flat layers. It follows a hyperbola whose curvature depends on the distance to the multilayer from the surface z, as well as on the distance between the source and the receiver x. This phenomenon arises from the fact that the propagating wave is spherical, meaning that the time it takes for the wave to hit the reflector and reach the receiver is different for each of them. Given a horizontal reflector, the further the station is from the source, the longer it takes. Mathematically, the reflected time traduces into the following hyperbolic expression:

$$t_r = \frac{\sqrt{x^2 + 4z^2}}{v} \tag{5.4}$$

This geological effect is known as normal moveout (NMO). A more detailed explanation can be found in Chapter 7. For this simulation, the multilayer is located at a certain distance from the surface. Each layer in the multilayer is around 7.23 m big each. The deeper it is located, the less the curvature the data suffers, but the longer the simulation takes. There is this trade-off between the simulation time and the precision dt we can afford. Additionally, the further apart the stations are, the more of this curvature is captured.

NMO correction is then another step in our preprocessing. Section 7.1 shows how to implement it, for which the velocity model v is needed. However, one limiting problem is that NMO correction, as it was implemented, works up to some curvature. When NMO is too strong, its conventional correction fails. It turned out to be a shortcoming in, say, cases

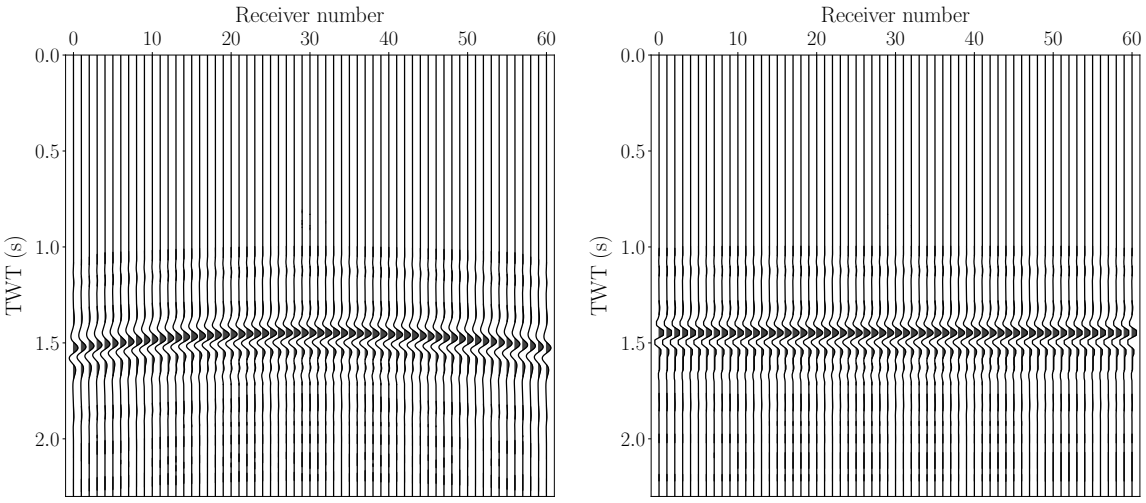
where a rather smaller first layer is set in order to have a shorter simulation time, thus entailing a greater curvature. More sophisticated implementations that I was not aware of may have performed well. Instead, some time was spent in implementing an intriguing non-conventional method to correct for normal move-out.

The idea of this alternative method comes from (Chen *et al.*, 2018), and it is based on Dynamic Time Warping (DTW), a common practice in signal processing. Such method looks into the similarity between two time series $\mathbf{x}$ and $\mathbf{x}'$, and matches points up from both series (x_i, x'_j) which are deemed similar. The sequence of all the similarity pairs is referred to as *path*, and it serves as a bridge to group time series together. Overall, if two time series are highly similar, then they belong in the same category. The differences themselves between time series can be due to any reason, e.g., a time perturbation. Appendix B is entirely about DTW and how it can be implemented.

DTW can be deployed to correct for NMO. It relies on the assumption that two adjacent seismic traces are similar. We start off with the zero-offset trace $\mathbf{s}_0$, which is our middle trace. This one remains intact, as it does not have NMO, and serves as a reference for the two adjacent gathers $\mathbf{s}_{-1}$ and $\mathbf{s}_1$. Next, DTW calculates the similarity path between two adjacent traces, say, $\{(s_0^i, s_1^j) \mid 1 \leq i, j \leq N_t\}$ for $\mathbf{s}_0$ and $\mathbf{s}_1$. The similarity path happens to contain the information about the time shift $|j - i|$, so NMO can be compensated by setting $s_1^j = s_1^i$ for all the (i, j) pairs. The trace $\mathbf{s}_1$ is now corrected, and becomes the reference for the next adjacent trace $\mathbf{s}_2$. This process goes on cumulatively between contiguous traces until they are all corrected.

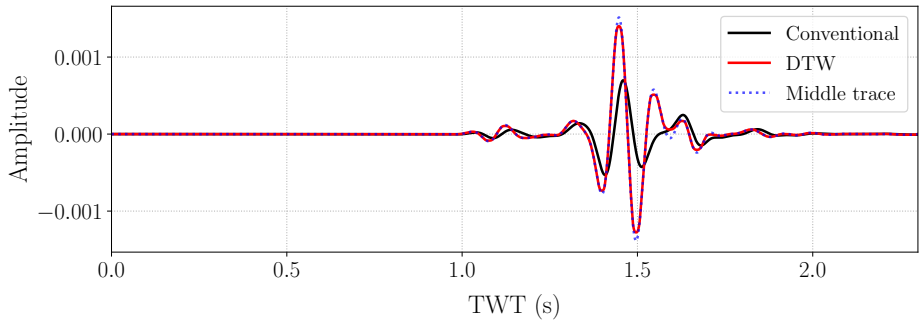
DTW approach differs greatly from the conventional method, as it does not look into reflected times and, hence, it does not need a velocity model. Figure 5.3 exhibits the results of both methods when they are applied to Figure 5.2b. The conventional NMO correction on the left, and the DTW approach on the right. We notice the conventional method has troubles correcting the time shift for larger offsets. For its part, the DTW method outperforms the conventional one, and goes around the problem of far stations. Stacking is exhibited in Figure 5.3c for both methods. It is clear that DTW gives a closer match to the middle trace, while the conventional approach distorts the waveform.

Conventional NMO correction usually stretches out the amplitude of seismic waveforms, as it is shown in section 7.1. The degree of stretch increases with increasing offset. DTW goes around this problem as it has got the zero-offset trace as reference. However, NMO correction does an excellent job improving the quality of the data when noise is present. One of the problems encountered when employing the DTW tool is that the noisy amplitudes are rounded when the number of samples in the data is not large enough. Or when contiguous traces are not close enough, so their similarity drops. The original shape of the wave would not be fully preserved. Computation wise, it becomes long and heavy. So, in the end we would like to restrict ourselves to scenarios under which we know that the conventional method works.



(a) Conventional NMO correction.

(b) NMO correction using DTW.



(c) Stacking data from (a) and (b).

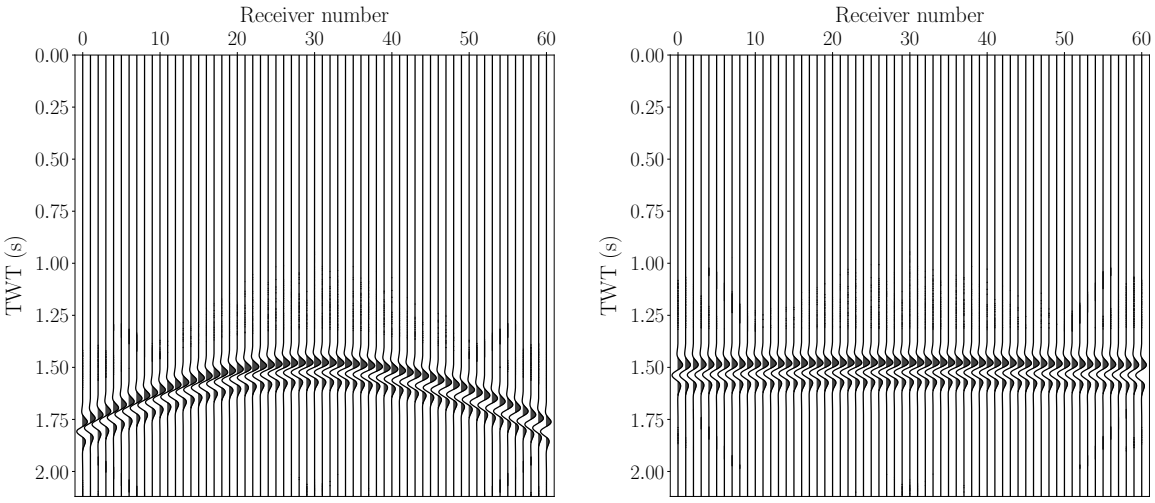
Figure 5.3.: NMO correction using conventional method (a) and dynamic time warping (b). (c) Stack of all the gathers after correction.

5.4 Two layer model

We shall start off using the simplest geological setting: a horizontal two-layer model. The purpose is to see what the data looks like when the simulated wave meets a single interface, and whether it is compatible with the convolutional model or not. If so, we proceed with the multilayer, which, in the end, is a set of horizontal interfaces.

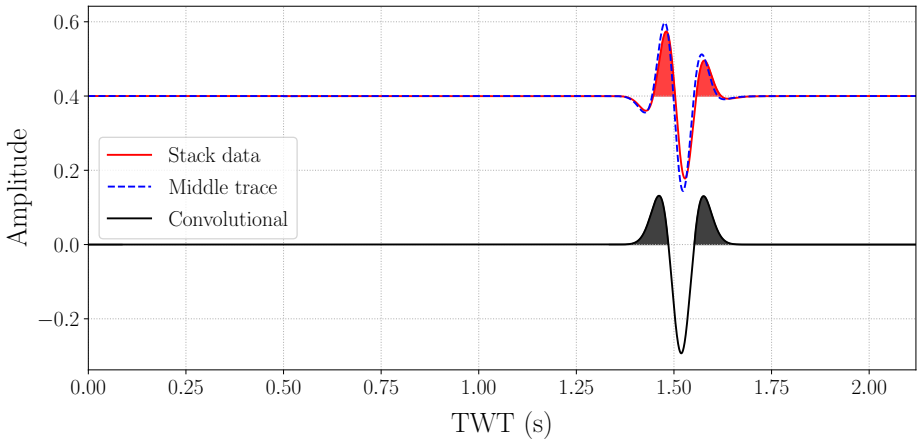
Figure 5.4 shows the outcome of a simulation free of noise. The interface is 1200 m under the surface. Above it, the velocity is $v_p = 1800\,\mathrm{m/s}$, and, below, $v_p = 1200\,\mathrm{m/s}$. Once more, Figure 5.4a proves the need to correct for NMO. The conventional NMO correction is chosen and exposed in Figure 5.4b. Next, the gathers are stacked together, resulting in the red profile of Figure 5.4c. For the shake of comparison, the zero-offset trace in the CMP is plotted on top too (dashed blue line). Although there is no noise and, hence, they should look the same, we notice they are slightly different. This is because NMO alters the frequency content and stretches the data waveform out. However, it does not seem to be a big problem here.

Figure 5.4c depicts in black the result of convolution, i.e., the seismogram computed using equation (3.3). Note that both the stack and the convolutional traces have been shifted



(a) Muted data.

(b) Conventional NMO correction.

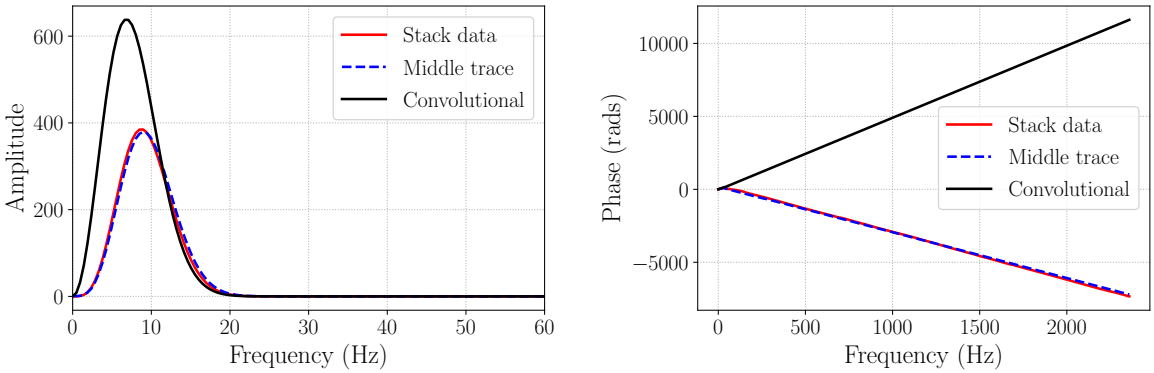


(c) Data.

Figure 5.4.: Results of the noise-free horizontal two-layer simulation. In (c)-black, the seismic trace computed using convolution $\mathbf{s}_{\text{conv}}$. In red, stacked data after NMO correction ($\mathbf{s}_{\text{stack}}$). The dashed blue profile represents the middle trace of the muted CMP.

apart for better visualization. At first sight, it seems like SPECFEM introduces some kind of phase rotation to the data with respect to the convolutional profile. Nonetheless, the convolutional model is an overly simple model, whereas the gathers capture the real behavior of a wave being reflected, so differences must be expected.

Let us look into the frequency content of each signal. The amplitude and phase spectra can be visualized in Figure 5.5. We see that differences in frequency amplitude between the stacked data and the zero-offset trace are rather negligible. Additionally, SPECFEM shifts slightly the dominant frequency to the right, compared to the convolutional data. This traduces into a shorter duration of the pulse. It is noteworthy the fact that, since there is only one interface (only one delta-like reflection coefficient), the shape of the data waveform follows that of the wavelet. This is why the spectra have then same form as that of Ricker wavelet. Phase wise, the convolutional model diverges from SPECFEM. This clearly reflects



(a) Amplitude spectra.

(b) Phase spectra.

Figure 5.5.: Noiseless two-layer model. Spectra of all the signals: stack, zero-offset and convolutional data. Note that the 2π radian cycles are sequentially stacked up together, so that there is a single increasing/decreasing graph for each plot.

some kind of phase rotation in SPECFEM, as expected by looking at the shapes in the time domain.

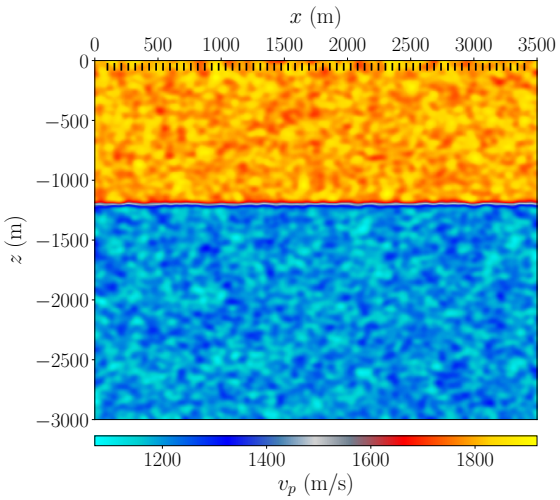
Overall, the convolutional model and SPECFEM produce signals compatible with each other, making the former one a valid approach.

We shall now add some noise to see if all of the above is preserved. White Gaussian noise is added to the velocity model, followed by a Gaussian kernel which correlates neighboring noisy points and smoothes the final landscape out. Figure 5.6a exhibits the noisy velocity model. As expected, the otherwise perfectly horizontal boundary is now rather bumpy, which makes it a near-horizontal reflector. Note that, this way, noise is introduced into the reflectivity, and, hence, convolved with the wavelet. The recorded signals after muting and NMO correction are included in Figure 5.6b, which results in the stacked trace of Figure 5.6c. Stacking helps reduce the signal-to-noise ratio, improving the quality of the data. Due to the noise, the NMO stretching is a bit larger though. Despite this, the shape of the wavelet is conserved and comparable with the convolutional one. The dashed magenta line represents the noiseless convolutional trace. The noise in the reflectivity causes it to not be a delta function anymore and it is consequently broadened. Then, when it is convolved with the wavelet, the resulting trace waveform adopts a slightly different shape. This is why the noisy and the noiseless data do not follow the same waveform, but similar. If the noise was only introduced in the trace free of noise, both would have the same base wavelet form.

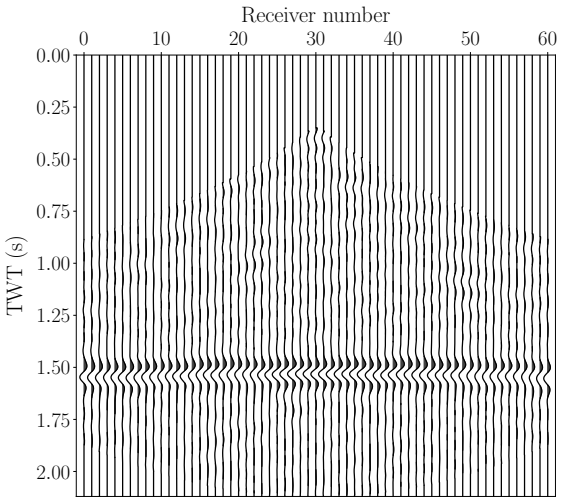
All in all, we can say that the convolutional model is valid approximation for a single interface, as the results show comparable behaviors, with a phase rotation off. Let us see now if the convolutional approach is able to capture the multiple reflections that occur inside the multilayer.

5.5 Multilayer model

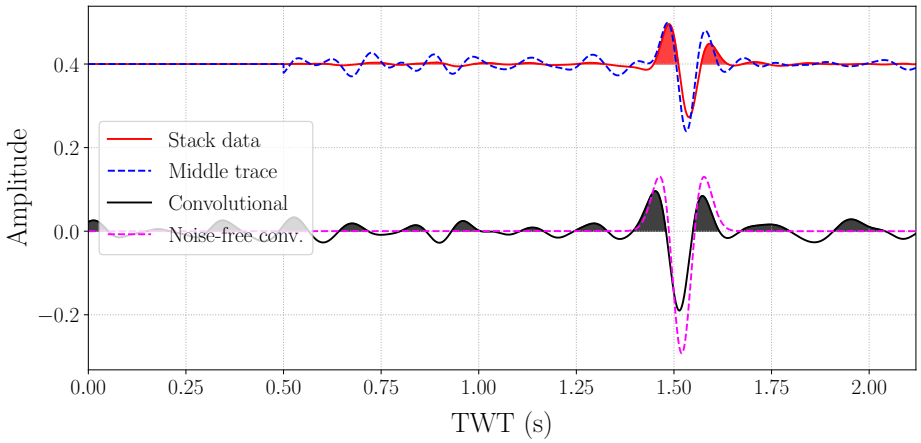
The test model looks like the one depicted in Figure 5.2. The structure entirely consists of horizontal layers, which means no dependence on x: $v_p = v_p(z)$ The first layer is 1500 m,



(a) Noisy v_p model.



(b) Data after muting and NMO correction.



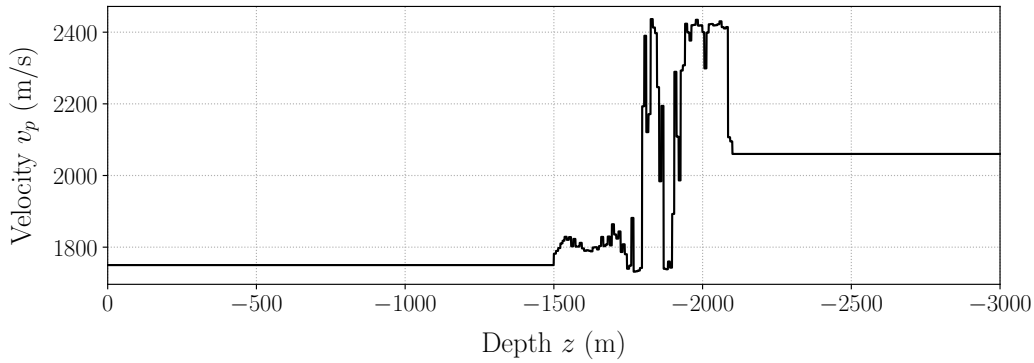
(c) Data.

Figure 5.6.: Results of the noisy horizontal two-layer simulation. (a) The velocity model after adding Gaussian noise. (b) Seismic data after muting and NMO correction. (c) Data stacked (red), the zero-offset trace (dashed blue), the noisy convolutional trace (black) and the convolution free of noise (dashed magenta).

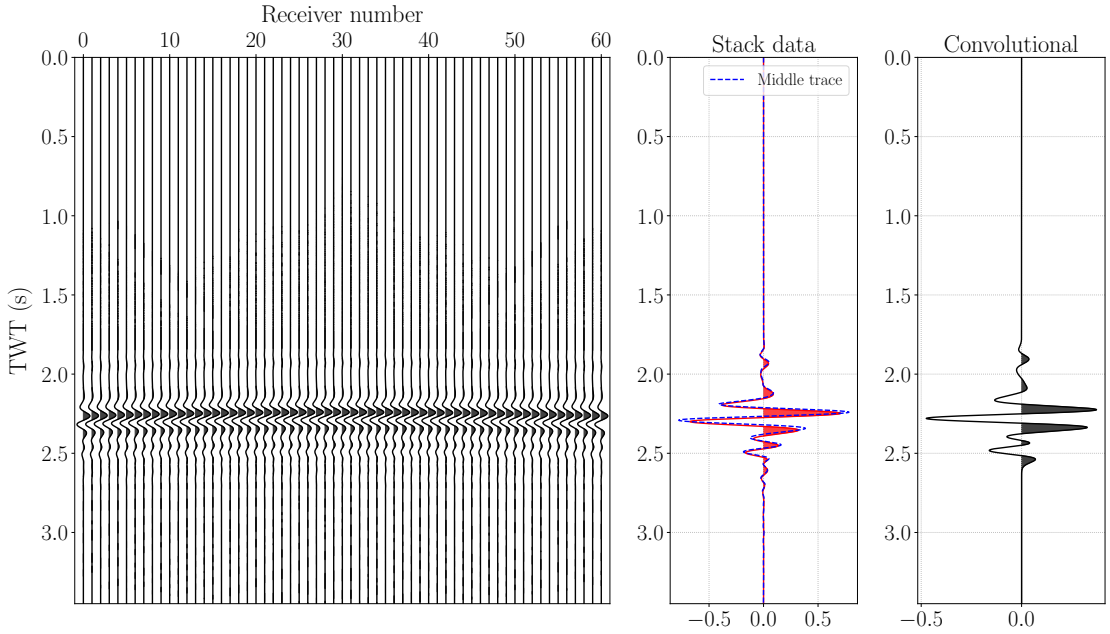
followed by 83 thinner layers of about 7.23 m each. The velocity profile is exposed in Figure 5.7a. This is the noiseless scenario, which is always a convenient starting point. Figure 5.7b summarizes the outcome of the simulation.

The resemblance between the stacked trace and the convolutional trace throws some light to our approach. Namely, the convolutional model is capable of reproducing the real data to a high degree. We see that the stacked data showcases a more wiggly tail than the convolutional model. This is attributed to the presence of echos or multiples, which are convoluted signals that root in the inner reflections that occur inside the layers.

Figure 5.8 shows the spectra of our three recurring signals: stacked, zero-offset and convolutional. The frequency content had to be re-scaled so it could be compared with the spectrum of the convolutional trace. The observations are twofold: firstly, the Ricker wavelet-like envelope is conserved for all the signals and, secondly, NMO reduces a bit the



(a) Velocity v_p profile.

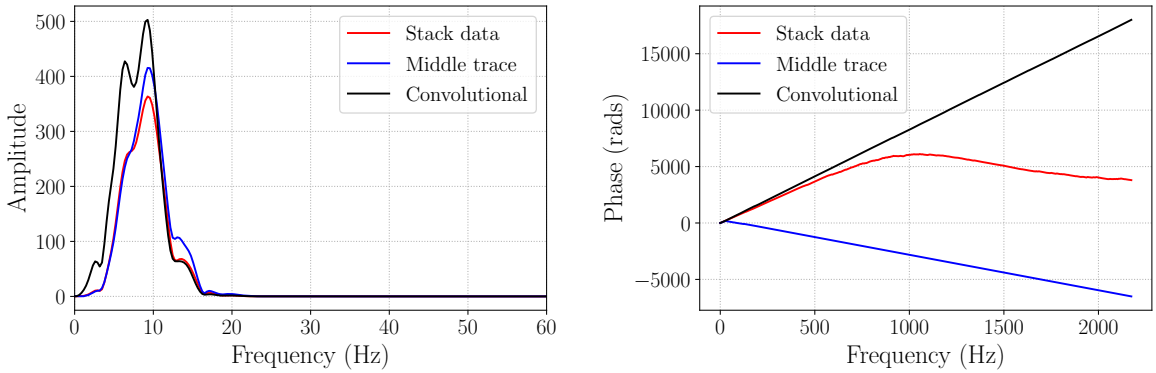


(b) Simulation results.

Figure 5.7.: Simulation free of noise. (a) 1D velocity profile: $v_p = v_p(z)$. (b) Results: data after muting and NMO correction, together with the stacked trace and the convolutional model. Mention that the stacked data has been rescaled for better visualization.

frequency with respect to the middle trace. On the other hand, the phase tells us about the time transformation that the wavelet may experience. In the two-layer case, we could see a clear phase shift between data and convolution from the very beginning. Here, for the initial small frequencies the data (red) follows convolution's phase (black), then for high frequencies it takes a turn in direction of the middle trace (blue). So, stacking seems to make up for the time shift in a large scale, but not for events that occur in a smaller time scale. It could be that these latter events have their origin in the multiples.

As a whole, the convolutional model preserves the main features of the real data in a multilayer. We shall add noise in order to be a bit closer to a real situation. This is exhibited in Figure 5.9. Again, white Gaussian noise is added and convolved with a 2D Gaussian kernel to smooth it out and correlate points, which results in the velocity model



(a) Amplitude spectra.

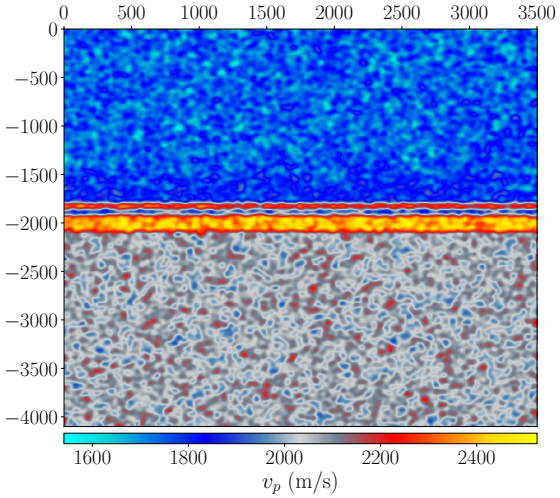
(b) Phase spectra.

Figure 5.8.: Noiseless multilayer. Spectra of all the signals: stack, zero-offset and convolutional data.

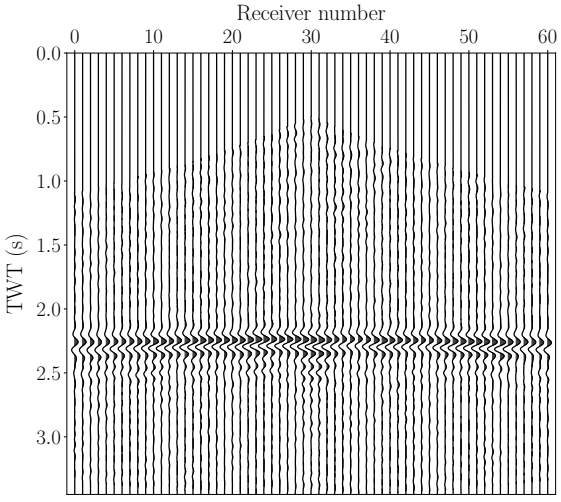
of Figure 5.9a. The small changes between thin horizontal layers are no longer present or well defined, but only the significant ones. Either way, a Ricker wavelet with frequency of $f_0 = 7.0\,\text{Hz}$ captures up to $2/f_0 \simeq 0.28\,\text{s}$ of precision. So, a layer about $l = 7.23\,\text{m}$ big with velocity $1800\,\text{m/s}$ traduces into a TWT $2l/v_p \simeq 0.008\,\text{s}$, which is two factors smaller than the time scale the wavelet is sensitive to. This means that the single reflection events are not relevant for the wavelet, but an average of a couple of them.

Figure 5.9b includes the data gathers after muting and NMO correction, which are stacked resulting in the red waveform of Figure 5.9c. On top, the dashed blue line, we see the middle (zero-offset) trace. It is clear that stacking reduces the noise impact, thus improving the signal-to-noise ratio to a great extent. The black data represents the noisy convolutional model, which, again, reproduces the real data fairly well. For comparison purposes, the dashed magenta line shows the noiseless counterpart of the convolutional trace. Once more, even when noise is added to the reflectivity, the overall seismogram waveform remains because the wavelet is only sensitive to large events on average that stand out in the noise sea.

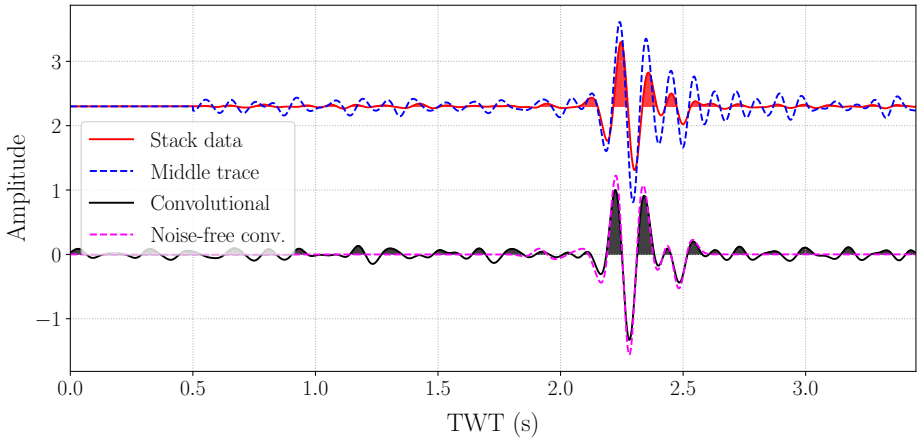
In short, the convolutional model is able to reproduce data to a valid degree. So, we would be in a position to use convolution as the forward model for inversion.



(a) Noisy v_p model.



(b) Data after muting and NMO correction.



(c) Data.

Figure 5.9.: Results of the noisy multilayer. (a) The velocity model after adding Gaussian noise. (b) Gathers after muting and NMO correction. (c) Data stacked (red), the zero-offset trace (dashed blue), the noisy convolutional trace (black) and the convolution free of noise (dashed magenta).

Chapter 6.

Inversion of simulated data

Given the magnitude of the foregoing velocity models, the convolutional model seems to account for the real data. This makes convolution a candidate forward model for inversion, as explained in section 3.4. It is appropriate to inquiry how will the convolutional model take in the real behavior of the data.

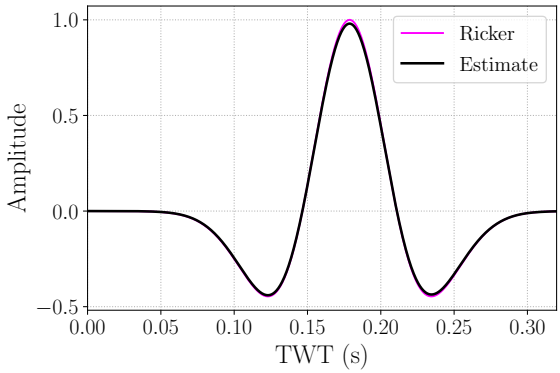
6.1 Two-layer model

Starting off with the ideal noiseless data from Figure 5.4c, we will apply inversion to estimate the wavelet $\hat{\boldsymbol{w}}$ from both the convolutional ($\mathbf{s}_{\text{conv}}$) and the stacked ($\mathbf{s}_{\text{stack}}$) traces. The results are shown in Figure 6.1. Due to the simplicity of this inversion scenario, the weighting matrix $\mathbf{W}$ from the Tikhonov formulation was not required. The solutions have been all found through equation (3.15) with $\varepsilon = 0.1$. The estimated wavelets from both convolution ($\hat{\boldsymbol{w}}_{\text{conv}}$) and stacking ($\hat{\boldsymbol{w}}_{\text{stack}}$) are shown, respectively, in Figures 6.1a and 6.1b. In both cases, the Ricker wavelet is plotted too (pink line), as it is the theoretical wavelet used in both cases.

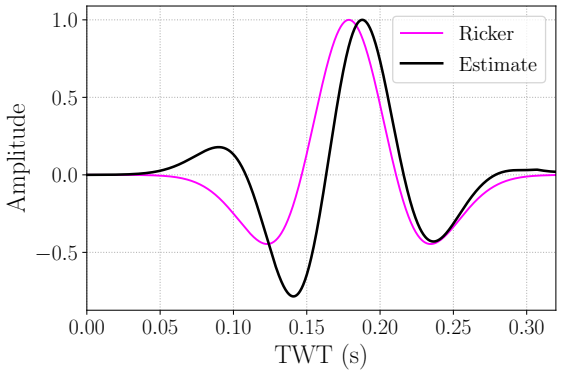
The shape of $\hat{\boldsymbol{w}}_{\text{conv}}$ coincides perfectly with the theoretical Ricker wavelet. This is expected since the convolutional trace itself is synthetically produced by convolving the Ricker wavelet with the single reflection coefficient. There is no noise hindering this convolution operation, so Tikhonov correctly estimates the wavelet. This also gives a perfect match when estimating the data $\hat{\mathbf{s}}_{\text{conv}} = \mathbf{R}\hat{\boldsymbol{w}}_{\text{conv}}$, as shown in purple in Figure 6.1c. Right on top, with a dashed black line, the true convolutional data $\mathbf{s}_{\text{conv}}$ is plotted.

Conversely, the estimated wavelet off the stacked data $\hat{\boldsymbol{w}}_{\text{stack}}$ looks like a phase rotated version of the theoretical Ricker wavelet. This aspect was already spotted in the discussion of Figure 5.4c, due to the difference between the stacked and the convolutional data. So, it seems like the non-linearities that occur in real life are warped up within the convolutional frame through a phase rotation in the wavelet. So, when this time perturbed wavelet is convolved with the reflection coefficient, it reproduces perfectly the original, real data. Note that the resultant wavelets in Figures 6.1a and 6.1b have been re-scaled to have unit amplitude.

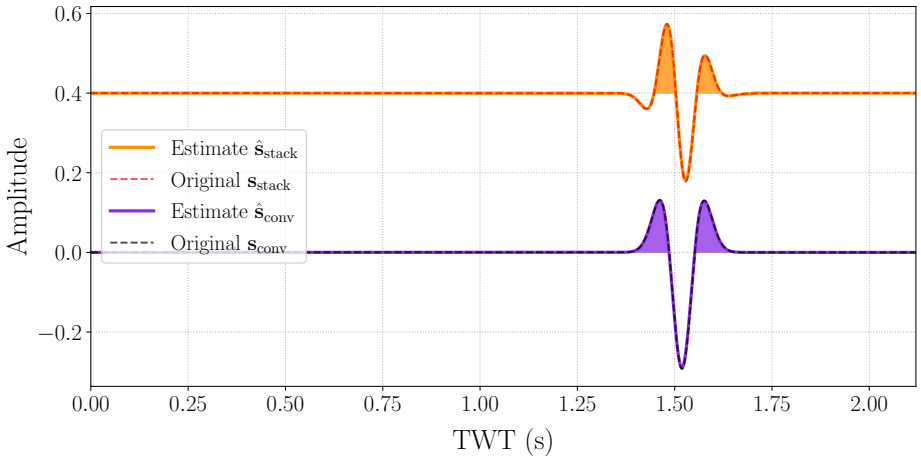
Mention also that the depicted wavelets are a time window of the full wavelet, which is all zeros for the rest of its duration until the end of the simulation. Theoretically, the Ricker wavelet, as it is defined in equation (3.11), is a zero-phase wavelet. Yet, here we treat it as a mixed-phase wavelet. This is, a wavelet that is zero at time zero, but whose maximum



(a) Wavelet estimate from convolution $\hat{\boldsymbol{w}}_{\text{conv}}$.



(b) Wavelet estimate from stacking $\hat{\boldsymbol{w}}_{\text{stack}}$.



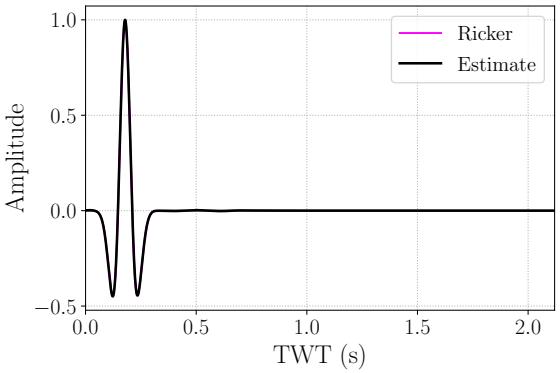
(c) Estimated seismic data.

Figure 6.1.: Results of inversion on the two-layer model. Not-weighted Tikhonov equation (3.15) used for inversion. (a) Wavelet estimate from convolutional trace. Note that it has been rescaled to unit for better comparison with the Ricker wavelet (pink). (b) Wavelet estimate from stacked trace. (c) The data estimates using the convolutional model as the forward model $\hat{\mathbf{s}} = \mathbf{R}\hat{\boldsymbol{w}}$.

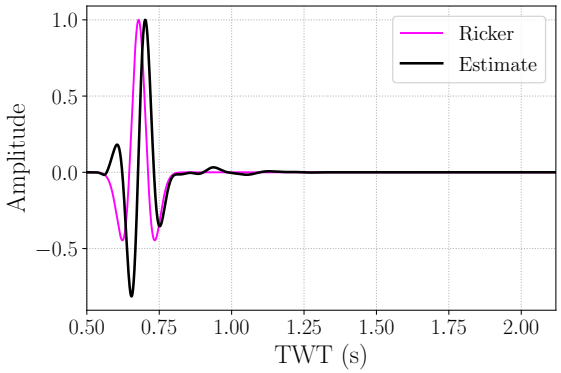
energy is located in the middle of its duration. It can be thought of as a theoretical Ricker wavelet that has been fully shifted to the range of positive times. Here, the wavelet starts to make sense at the moment when the simulation starts, and not anymore when the simulation finishes. This is why the wavelet is defined within the whole positive simulation time, but only the Ricker-like pulse of energy at the beginning contributes. Thus, compact support is fulfilled.

Tikhonov regularization works fine with the clean, ideal data. Let us now jump to the noisy scenario exposed in Figure 5.6c. Here, the simplicity constraint of the solution based on the L_2 norm is not enough, and it requires the weighted Tikhonov regularization approach, i.e., equation (3.19). The noisy solutions found for both convolution ($\hat{\mathbf{s}}^{\mathbf{n}}_{\text{conv}}$) and stacking ($\hat{\mathbf{s}}^{\mathbf{n}}_{\text{stack}}$) are illustrated in Figure 6.2.

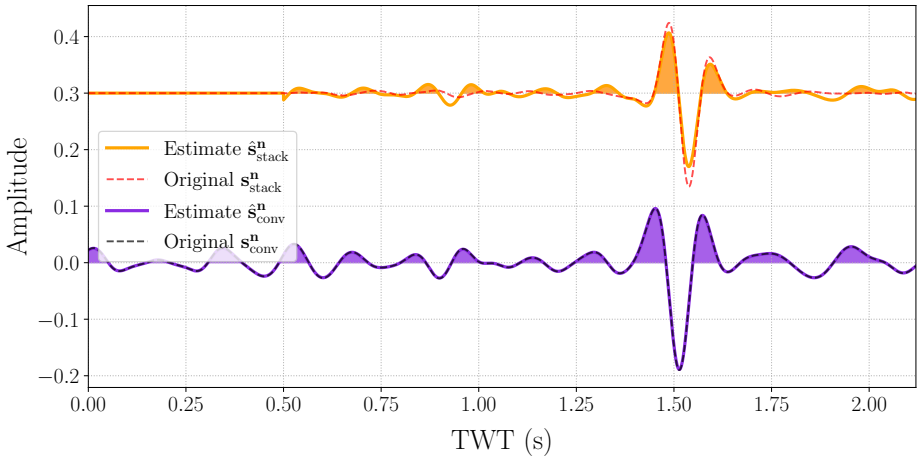
In section 3.4, it was said that one way of introducing prior information to our problem was through exponentially increasing weights, so the reverberating terms in the tail are highly penalized. This was carried out by defining the diagonal of the $\mathbf{C}$-matrix as an exponential



(a) Wavelet estimate from convolution $\hat{\boldsymbol{w}}^{\mathbf{n}}_{\text{conv}}$.



(b) Wavelet estimate from stacking $\hat{\boldsymbol{w}}^{\mathbf{n}}_{\text{stack}}$.



(c) Estimated seismic data.

Figure 6.2.: Noisy inversion results for the two-layer model. (a) Wavelet estimate from convolutional trace. Note that it has been rescaled to unit for better comparison with the Ricker wavelet (pink). (b) Wavelet estimate from stacked trace. (c) The data estimates using the convolutional model as the forward model $\hat{\mathbf{s}} = \mathbf{R}\hat{\boldsymbol{w}}$.

function $ar^{k \cdot i}$, where i is the position on the diagonal (not the complex number). Yet, the results obtained out of this weighting were still unfeasible and unstable. The estimated wavelets successfully fulfilled compact support, but sometimes they became very wobbly and doubtful at the beginning, with non-zero energy at time zero. For the wavelet to have physical meaning, thus retrievable through regularization, it must also meet causality, which entails zero amplitude at time zero. In the endeavor to find solutions in the model space that lived up to both causality and compact support, an additional decreasing exponential is added to the constraint when needed. It is aimed at penalizing the first few terms of the wavelet, forcing it to start at zero. This way, the weights along the diagonal of the $\mathbf{C}$-matrix follow the sum of two exponentials $a_1 r_1^{k_1 \cdot i} + a_2 r_2^{-k_2 \cdot i}$, with $k_1, k_2 > 0$.

The wavelet $\hat{\boldsymbol{w}}^{\mathbf{n}}_{\text{conv}}$ estimated off the convolutional trace (Figure 6.2a) was obtained without the need of the decreasing exponential ($a_2 = 0$), and with parameter values $a_1 = 1.0$, $r_1 = 1.035$ and $k_1 = 0.13$. The regularization parameter was set to $\varepsilon = 10$. We see how the tailing terms are killed, giving the wavelet compact support. The initial pulse of energy resembles the Ricker wavelet almost perfectly. Likewise, an almost perfect match

is then expected when estimating the data $\hat{\mathbf{s}}_{\text{conv}}$. Indeed, we appreciate how the original and the estimated convolutional traces match up altogether (purple plot in Figure 6.2c).

Let us comment now on the SPECFEM data $\mathbf{s}^{\mathbf{n}}_{\text{stack}}$. The estimated wavelet $\hat{\boldsymbol{w}}^{\mathbf{n}}_{\text{stack}}$ was extracted using $a_1 = 10.0$, $r_1 = 1.023$, $k_1 = 0.12$, $a_2 = 1.0$, $r_2 = 1.15$ and $k_2 = 0.3$. The regularization parameter is $\varepsilon = 10$. An exponential with more penalizing power was required to suppress wiggly high energy terms in the tail. Playing around with the parameters one always ends up with a wavelet which, again, shows a time perturbation with respect to the Ricker wavelet. Namely, it experiences a phase rotation that comes from the fact that the data is real, as discussed above. The data estimate $\hat{\mathbf{s}}^{\mathbf{n}}_{\text{stack}}$ (orange waveform) captures the real wave response when reflected through this time transformation of the wavelet. The solution does not get lost in the sea of noise.

The misfit of the real, stacking case is $E_{\text{stack}} = \|\mathbf{s}^{\mathbf{n}}_{\text{stack}} - \hat{\mathbf{s}}^{\mathbf{n}}_{\text{stack}}\|^2 \simeq 35$ when their amplitudes are suitably normalized so they become comparable with the convolutional case, where $E_{\text{conv}} \simeq 0.07$. Figure 6.3 shows the Z-score distribution of the residuals $r_i = \hat{s}_i - s_i$. In the convolution population (purple) more samples are contained within 1σ compared to the stacking population. The standard deviation of the former one is $\sigma_s \simeq 0.003$, and $\sigma_s \simeq 0.06$ for the latter one. A better performance with the noisy convolutional model was expected, but it also does a good job with the SPECFEM model.

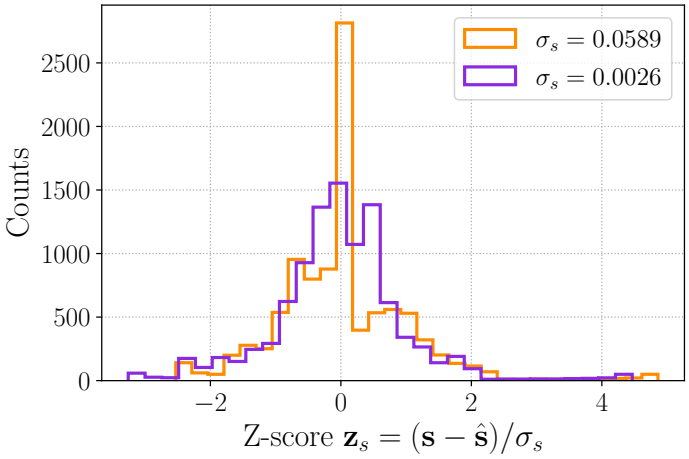


Figure 6.3.: Z-score distribution of the residuals, defined as the difference between the estimated and the real data.

6.2 Multilayer model

We now proceed likewise for the multilayer model. The noiseless results are obtained applying Tikhonov to the data $\mathbf{s}_{\text{conv}}$ and $\mathbf{s}_{\text{stack}}$ from Figure 5.7b. They are shown in Figure 6.4. In the convolution example, the decreasing exponential $\propto r^{-kx}$ in the weighting matrix $\mathbf{W}$ was still not needed to find a valid solution $\hat{\boldsymbol{w}}_{\text{conv}}$. It was obtained setting $a_1 = 1.0$, $r_1 = 1.01$, $k_1 = 0.09$ and $\varepsilon = 1.1$. Even tough it incurs into very minimal instabilities along the tail of the wavelet, the data estimate $\hat{\mathbf{s}}_{\text{conv}} = \mathbf{R}\hat{\boldsymbol{w}}_{\text{conv}}$ agrees perfectly with the original synthetic trace $\mathbf{s}_{\text{conv}}$ derived from convolution.

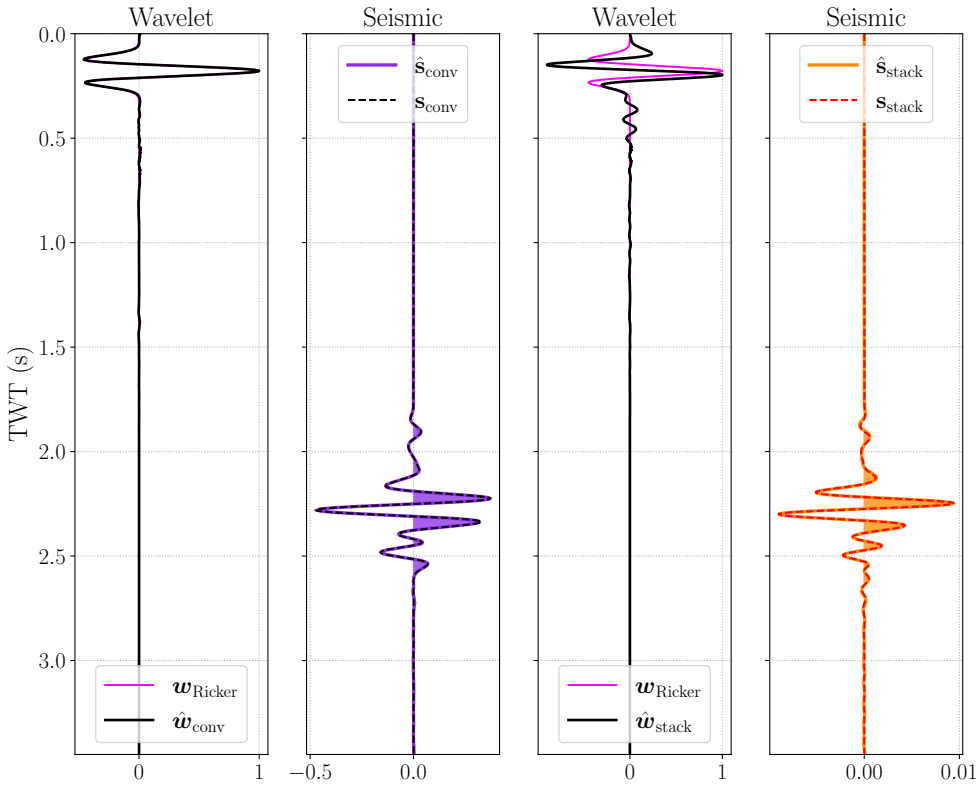


Figure 6.4.: Inversion results using noiseless data $\mathbf{s}_{\text{conv}}$ and $\mathbf{s}_{\text{stack}}$. From left to right: convolutional wavelet and seismic estimates, followed by the stacking wavelet and seismic estimates. All the plots include the estimates and the original/reference quantities.

For the stacked data, the two increasing and decreasing exponential weights were required to find a sensible solution. The parameters used were $a_1 = 1.0$, $r_1 = 1.015$, $k_1 = 0.07$, $a_2 = 1.0$, $r_2 = 1.15$, $k_2 = 0.5$ and regularization $\varepsilon = 1$. Again, the estimated wavelet $\hat{\boldsymbol{w}}_{\text{stack}}$ showcases a phase rotation with respect to the Ricker wavelet. The wiggly tail comes about due to the presence of echos that manifest in the data, but still it can be deemed a wavelet. The reproducibility of the data is optimal either way.

Let us now switch to the noisy scenario, which arises from the data in Figure 5.9c, and yields fairly good results too. They are exposed in Figure 6.5. Tikhonov still manages to come up with valid results. On one hand, the convolution solution matches the theoretical Ricker wavelet, except for a minimal reverberating tail due to the noise. It gives an estimate on the data that reproduces the synthetic trace almost perfectly. At this point one might question the overfitting of the noise. It happens that there are no discrepancies or approximations committed between how the data is synthetically produced and the actual forward model. Loosely speaking, the data are *forwarded* the same way they are *inverted*, so the perfect match is reasonable.

On the other hand, the SPECFEM solution, apart from the usual phase rotation, presents a very wobbly tail that roots in the noise on top of the non-linear multiples. In spite

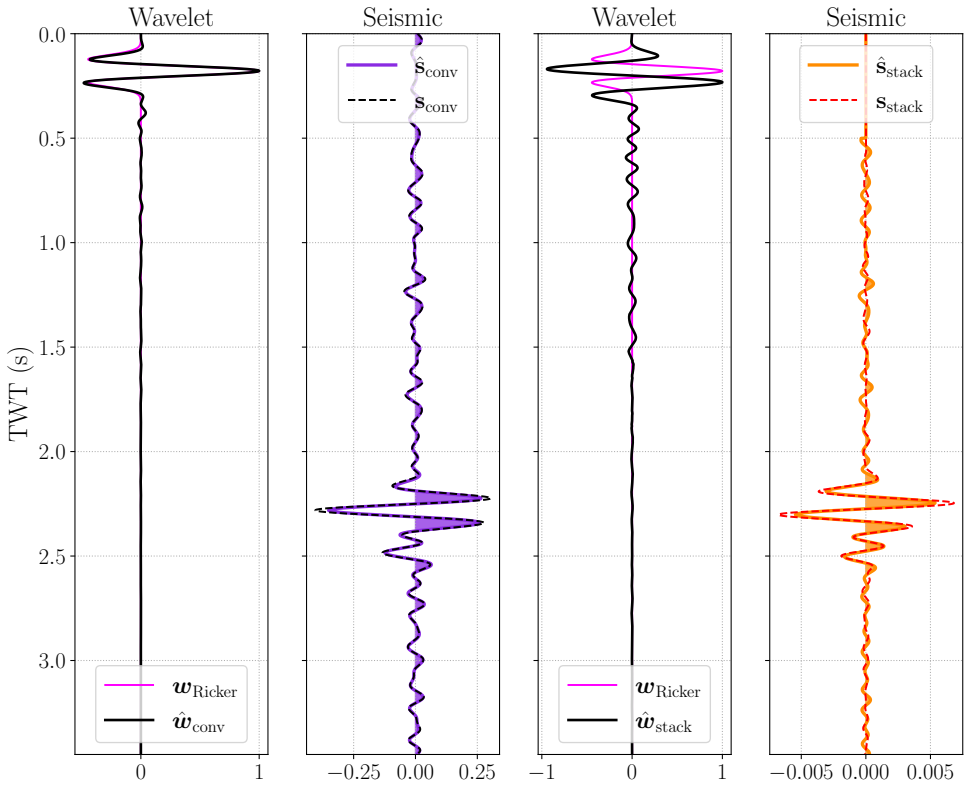


Figure 6.5.: Inversion results using noisy data $\mathbf{s}^{\mathbf{n}}_{\text{conv}}$ and $\mathbf{s}^{\mathbf{n}}_{\text{stack}}$. From left to right: convolutional wavelet and seismic estimates, followed by the stacking wavelet and seismic estimates. All the plots include the estimates and the original/reference quantities.

of the penalizing weights in $\mathbf{W}$, it is not physically possible to find a $\hat{\boldsymbol{w}}_{\text{stack}}$ with a less reverberating tail that, at the same time, fits the data. Again, it is all about this trade-off between the undetermined part of the wavelet controlled by its L_{weighted} norm and the misfit in the data estimates. We can only aim at a sensible wavelet-like solution which does not overfit the noise and still describes the main features contained in the data.

The residuals are included in Figure 6.6. For the distributions to be comparable in terms of spread, the signals have been re-scaled to the same order. The convolutional distribution shows an excellent match as envisaged, with the majority of its predictions within one $\sigma_s \simeq 0.02$. The misfit is $E_{\text{conv}} \simeq 9$. Conversely, the stacked distribution doubles the spread of the convolution $\sigma_s \simeq 0.04$. The misfit is $E_{\text{stack}} \simeq 25$, but everything within acceptable fitting values given that we are not interested in fitting the noise at all.

6.3 Increase non-linearities

The convolutional model is able to estimate the data fairly well given the magnitude of the reflection coefficients. But, up to what extent will the convolutional model be able to account for the non-linearities that occur in real life? This is the question we will attempt

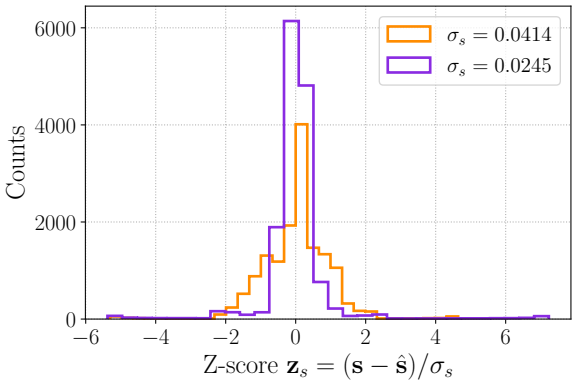


Figure 6.6.: Z-score distribution of the residuals, defined as the difference between the estimated and the real data.

to respond in this section. One way of increasing the non-linearity of the problem is to scale up the reflection coefficient series, resulting in stronger echos and, consequently, stronger attenuation or transmission losses. This can be achieved by getting sharper changes in the layered media. Since the convolutional reflectivity depends on the velocity, it suffices to scale up the velocity in a way that preserves the average velocity $\langle v_p \rangle$. Mathematically:

$$v_p^{\text{sc}} = \zeta \cdot (v_p - \langle v_p \rangle) + \langle v_p \rangle \tag{6.1}$$

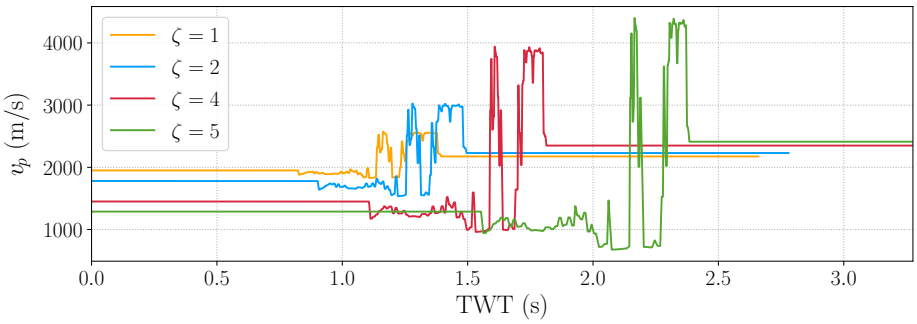
where v_p^{sc} is the scaled velocity model by a factor ζ. Figure 6.7 shows the noiseless velocity and reflectivity profiles for three values of $\zeta = 2, 4, 5$. A value of $\zeta = 1$ means no scaling, and one retrieves the same results and conclusions as before. We see that larger spikes in the reflectivity are obtained, specially at the beginning.

Proceeding the same way as before, we get the estimates shown in Figure 6.8. The results are obtained without any noise. For each subfigure, we find the wavelet estimate on the left and the data on the right. In black the true, original data, and, colored, the estimate. As we increase ζ more energy is accumulated in form of multiples along the tail in the real data. These echos are inherited by the wavelet estimate, so they are overfitted when it comes to predicting the data. In return, solutions lose the sense of wavelet as they present longer oscillations impossible to get rid of.

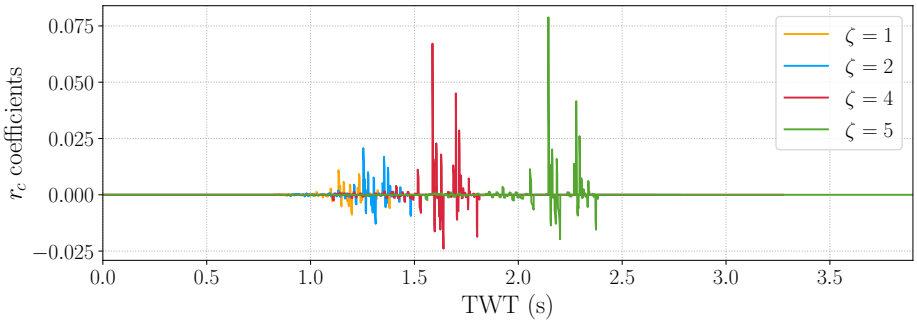
When the echos become too strong they mask the information about the structure that the reflectivity carries, and the estimated wavelet tries to fit them as part of the data. Other seismic inversion approaches must be employed in order to get such information out, where echos and other non-linearities do not hinder the whole perspective. So we have arrived to a clear bottleneck in our approach.

For the shake of completeness, let us include some noise in the $\zeta = 2$ case. The same conclusions apply, but for this case more restrictive weights had to be applied.

For all of the above, NMO has been corrected using DTW. As the non-linearities increase, it becomes harder for the conventional approach to correct the time shift in the reflected times.

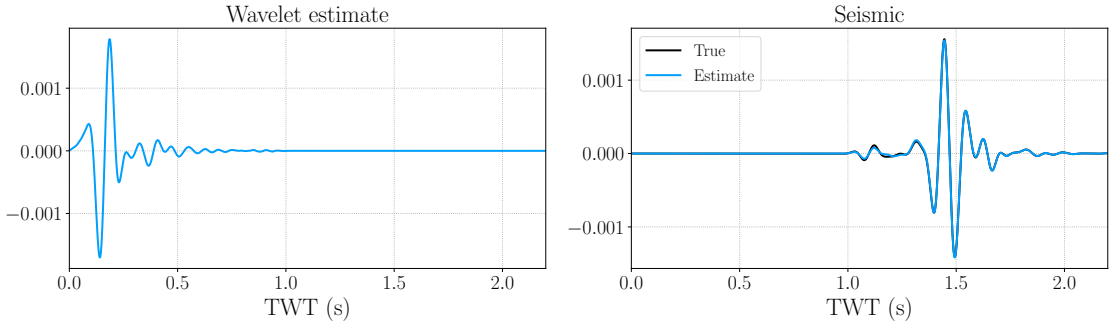


(a) Velocity profiles for 4 different scales: $\zeta = 1, 2, 4, 5$.

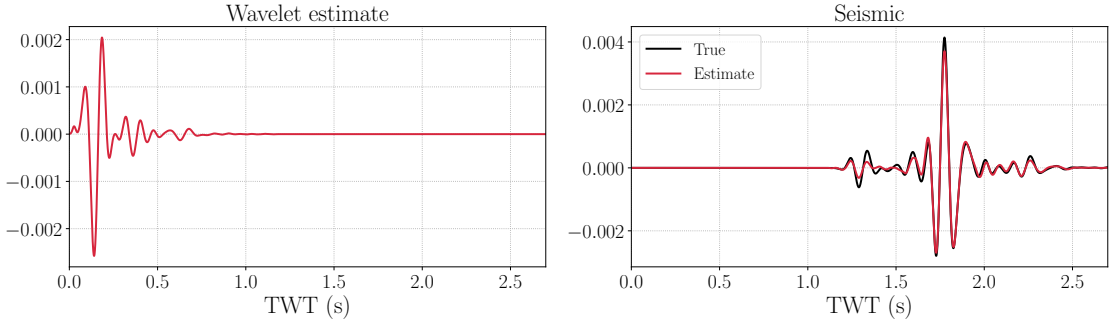


(b) Scaled reflectivity series.

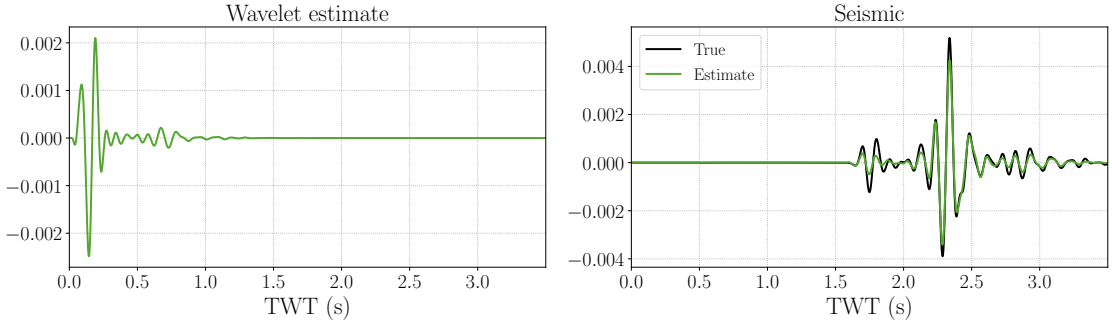
Figure 6.7.: (a) Scaled velocities profiles from $\zeta = 1$ (no scaling) to $\zeta = 5$, following equation (6.1). (b) The corresponding convolutional reflection coefficient series for the different scales ζ.



(a) Wavelet and data estimates for $\zeta = 2$.



(b) Wavelet and data estimates for $\zeta = 4$.



(c) Wavelet and data estimates for $\zeta = 5$.

Figure 6.8.: Noiseless estimates from the scaled profiles from Figure 6.7.

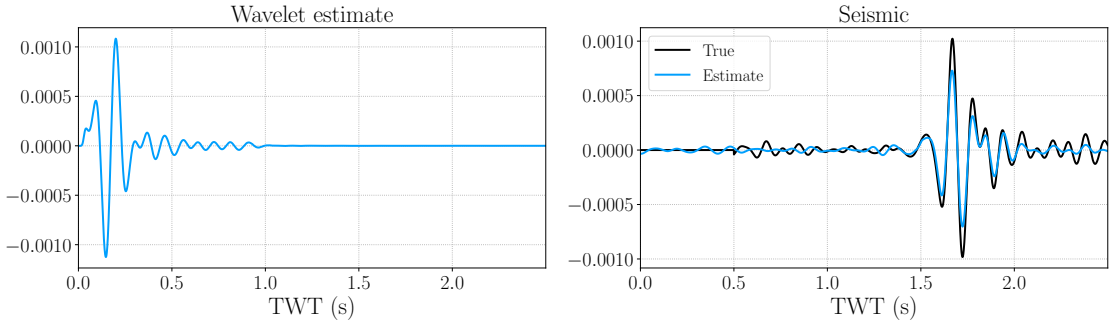


Figure 6.9.: Noisy wavelet and data estimates for $\zeta = 2$.

Chapter 7.

Data preprocessing

Data preprocessing is a must in every seismic processing procedure. In our case, the data is 2D and comes from simulating the propagation of spherical waves. The convolutional model, instead, is 1D and allows for normal reflections only, i.e., assumes that the waves are planar wavefronts. Over large distances, spherical waves can be approximated as planar. However, that would require considerably more simulation time, memory and disk space. Instead, we resort to NMO correction to correct the curvature. Finally, stacking renders the data amenable for our convolutional purposes. Moreover, we have seen that NMO correction and stacking improves the quality of the data when noise is present.

Additionally, the problem of seismic migration is also introduced. It was one of the late ideas to test in the project, but that, unfortunately, could not be carried all the way through. In essence, when the reflectors are tilted, the velocity and density models do not depend solely on the depth z anymore, but also on x. Migration effects are those that involve horizontal displacements, and lead to miscalculations if they are not corrected. Migration, in combination with NMO correction, was attempted with the purpose of using our convolutional approach.

7.1 NMO correction

Section 5.3 mentions that due to the spherical nature of the waves, the receivers do not receive the signal from a single source simultaneously, but accordingly to the reflection time it takes for the wave to travel from the source to the receiver, thus shaping a hyperbola. Let us show this graphically. For a near-horizontal reflector, the ray paths from a single source is shown in Figure 7.1. A single layer of depth z and velocity v is illustrated, where we distinguish two main rays: the direct wave (red) and the reflected wave (black). The direct wave follows the straight line $t_d = x/v$, perceptible in Figure 5.2b. It provides no information, it is just a wave travelling from the source to the receiver, so it can be muted. On the other hand, the reflected wave involves both the distance to the receiver x and the depth of the reflector z. The shape of this wave front is hyperbolic:

$$t_r = \sqrt{t_0^2 + \frac{x^2}{v^2}} = \frac{\sqrt{x^2 + 4z^2}}{v} \tag{7.1}$$

where $t_0 = 2z/v$ is the two-way-travel vertical time. It is the minimum time at which a reflection will be recorded. The time delay $\Delta t = t_r - t_0$ is the so-called normal moveout.

All of this is depicted on the right panel of Figure 7.1. By correcting for this time shift, the reflection events appear to arrive simultaneously to the receiver.

In order to implement NMO correction, for each two-way-travel time step t_0 we find its corresponding reflected time t_r, which depends upon the distance x to the receiver. The amplitude of the trace at t_r is found by sampling the trace $s(t = t_r)$. Finally, the correction is applied $s(t_0) = s(t_r)$. This must be done for all the gathers in the CMP. The pseudo-code in Algorithm 1 summarizes NMO correction.

Algorithm 1: Conventional NMO correction. Here, N_t is the number of time steps and N_{offsets} the number of offsets or gathers. For each time t_0 and gather s_i, the corresponding reflected time t_r is calculated (`reflection_time`). The function `sample_trace` samples $s(t)$ at t_r to get the amplitude of the signal.

Input: CMP gathers **s**, offset positions **x** and velocity model **v**
Output: NMO correction

for $n_t = 0, \ldots, N_t - 1$ **do**
  $t_0 \leftarrow n_t \times N_t$
  for $i = 0, \ldots, N_{\text{offsets}} - 1$ **do**
    $t_r \leftarrow$ `reflection_time`(t_0, x_i, v_{t_0}) `// Computes reflection time (7.1)`
    $A \leftarrow$ `sample_trace`(s_i, t_r) `// Amplitude of signal s(t_r)`
    $s_i(t_0) \leftarrow A$

If there are additional layers, then the seismic energy at each interface is refracted/reflected according to Snell's law. The energy no longer travels in a straight line, as showed in Figure 7.1, and, hence, the travel times are affected. For small offsets, the travel time curve is still approximately hyperbolic, but the velocity, which controls the shape of the curve, is an average velocity of the velocities of all the layers above the reflector. This is, RMS (root-mean-square) velocity v_{rms} (GeoSci.xyz, n.d.).

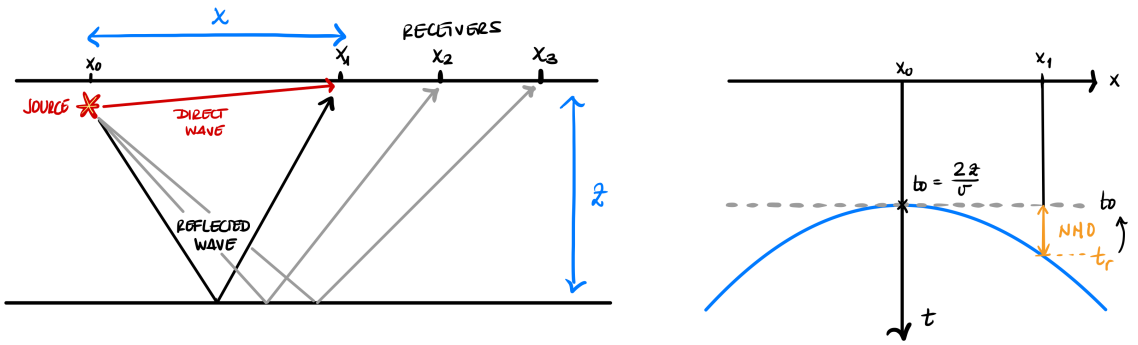
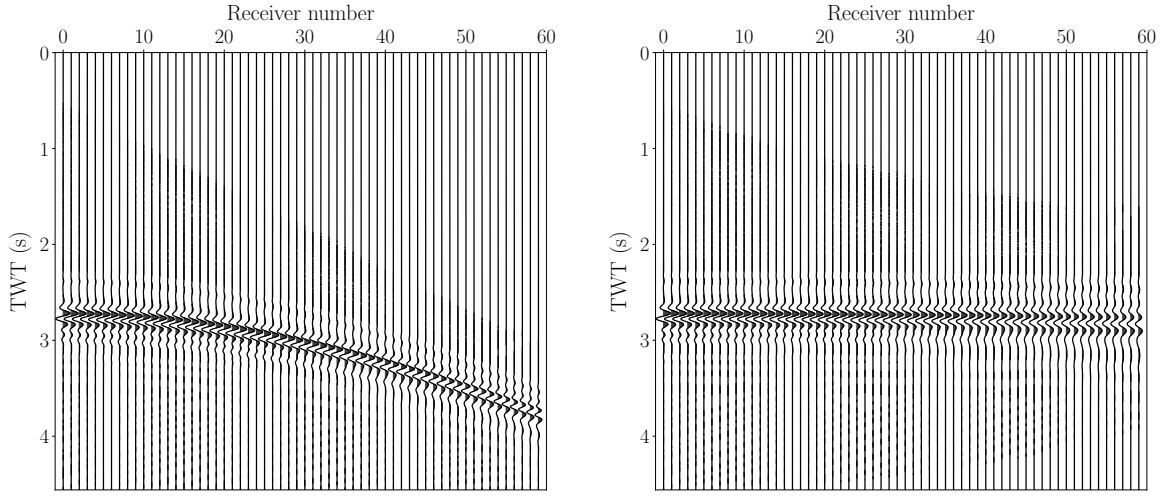


Figure 7.1.: On the left, raypath from source to receiver for a single layer of thickness z and velocity v. The receivers are located at positions $x_1, x_2, \ldots$. The zero offset gather is at x_0. On the right, reflected time diagram. In blue, the hyperbolic wave front. The NMO time shift is measured as $t_r - t_0$.



(a) Data before NMO correction. **(b)** Data after NMO correction.

Figure 7.2.: Normal move-out correction and the problem of stretching.

For each hyperbola:

$$t_r \simeq \frac{\sqrt{x^2 + 4z_i^2}}{v_i^{\mathrm{RMS}}} \tag{7.2}$$

where v_i^{RMS} is the RMS velocity for each layer of the N-layer model ($i = 1, \ldots, N$). The RMS velocity for the i^{th} layer is given by:

$$v_i^{\mathrm{RMS}} = \sqrt{\frac{\sum_{k=1}^{i} v_k^2 \tau_k}{\sum_{k=1}^{i} \tau_k}} \tag{7.3}$$

One of the problems with NMO correction is that stretches the waveform out, as we perceive in Figure 7.2, thus decreasing the frequency content of the wavelet. However, it does not seem to be a big problem within the first 30 offsets. Figure 7.2 shows a total of 60 offsets evenly distributed in 3500 m. Throughout the thesis, the offsets are laid out in the same way, but with the shot point in the middle, so that there are 30 receivers on either side.

7.2 Migration

Migration is the process of reconstructing a seismic section so that the reflection events are repositioned under their correct surface location and at a corrected vertical reflection time. In the project dipping reflectors were addressed in their simplest version: tilted boundaries; which results in a geometric displacement of the data. So, migration must be applied to place them in their true spatial position rather than at an assumed point in depth between the source and the receiver.

Figure 7.3 illustrates why the seismic section does not capture the true position of the events when it comes to dipping reflectors. The points x_1, x_2 and x_3 are the shot and

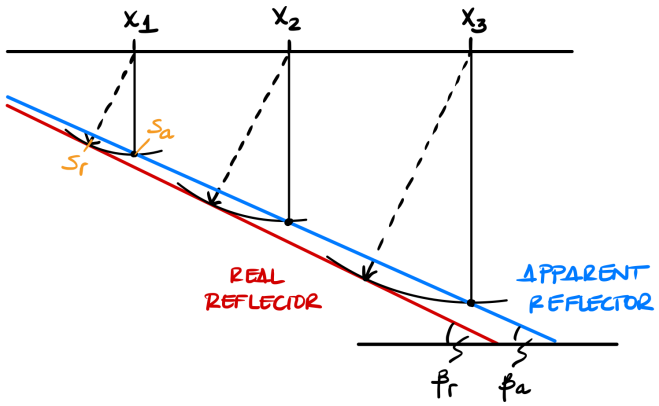


Figure 7.3.: Diagram showing the raypath for a zero-offset reflection from a dipping reflector and the resultant apparent dip. The angles β_r and β_a are, respectively, the angles of the real and apparent reflectors with respect to the horizontal. The point s_a is the apparent point one would observe before migration, while s_r refers to the real position obtained after migration.

receiver points on the surface. In the real case scenario (red in Figure 7.3), the spherical wavefront of the seismic wave will encounter the real dipping reflector at s_r after some time t, then the reflected wave will travel back to, say, x_1. Distance-wise, it traduces into $d = vt$, or $d = vt/2$ if the two-way total travel time is measured, being v the velocity in the medium. Yet, in the data we would see the point s_a, which is s_r but shifted along the wavefront up to the vertical position right below the shot point. Such a point s_a is the same distance d from x_1 as s_r, since they lay on the same spherical wavefront (circular for 2D). If we now interpolate all the s_a points from all the shot points x_i, we get the apparent dip (blue in Figure 7.3). So, in the end of the day, the role of migration is to send the points laying on the apparent dip back to their real positions.

The *migrator's equation*

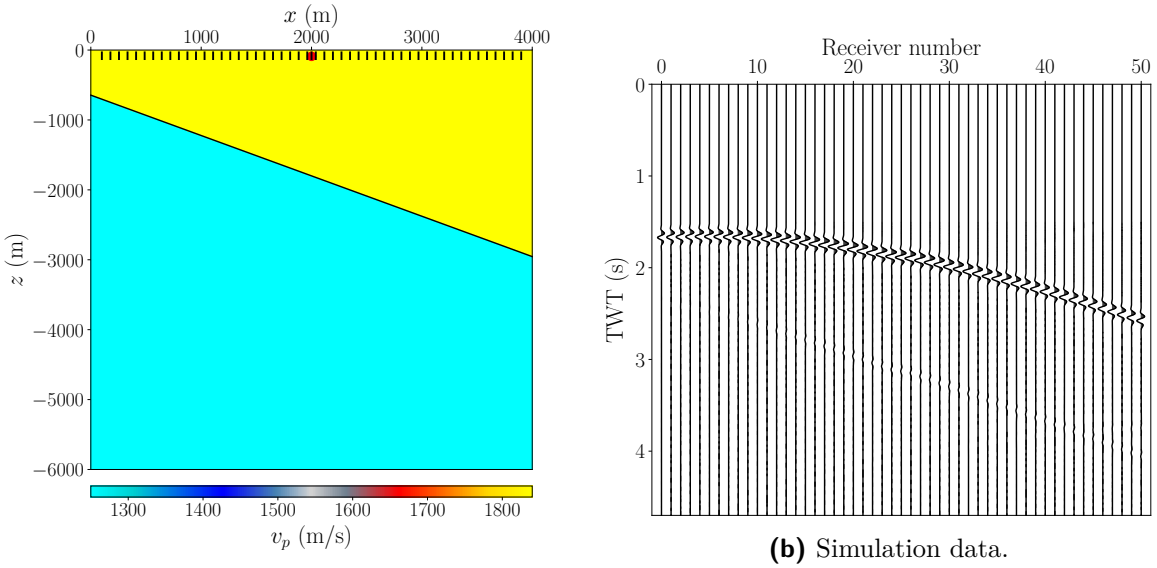
$$\tan \beta_a = \sin \beta_r \tag{7.4}$$

relates the angle of the real reflector β_r with the apparent dip β_a.

Figure 7.4 shows an actual simulation on a very simple setting with only one dipping reflector. On the left panel, what the velocity model looks like. On the right, we see that the tendency of the wiggles bend down. In an otherwise horizontal case, there would be a hyperbola with its maximum in the middle (assuming the shot is placed in the middle x-wise). The dip makes this maximum deviate upwards.

Zero-offset data is important to a geophysicist because the migration operation is much simpler, and can be represented by spherical surfaces. When data is acquired at non-zero offsets, the sphere becomes an ellipsoid and is much more complex to represent (both geometrically and computationally).

Migration can be broadly split into two fundamental types of migration, defined by the domain they are applied: time migration and depth migration.



(a) Velocity v_p model.

(b) Simulation data.

Figure 7.4.: Results from running the simulation on a simple two dipping layers geological setting. (a) The v_p model. The markers along the surface are the position of the receivers. The upper layer (yellow) has $v_p = 1840\,\mathrm{m/s}$, and the lower one (blue) $v_p = 1250\,\mathrm{m/s}$ (b) Seismic data.

- **Time migration**. It is applied to seismic data in time coordinates. This type of migration makes the assumption of only mild lateral velocity variations.
- **Depth migration**. It is applied to seismic data in depth coordinates, which must be calculated from seismic data in time coordinates. If lateral velocity gradients are significant, we need to use depth migration.

Adding now the possibility of having pre- and post- stacked data, four combinations can be made when applying migration: post-stacked time migration, post-stacked depth migration, pre-stacked time migration and pre-stacked depth migration. Plenty of methods are available for each scenario, but in principle we are interested in those that apply to pre-stacked time migration. Although our geological setting is not complex, pre-stacked methods are preferred as the complexity of the structure increases at the expense of higher computational costs.

Chapter 8.

Conclusion

During this thesis we have exploited deterministic inversion tools for wavelet estimation. Firstly, on 1D borehole data, where we saw the need of regularization. The naïve solution was unstable against noise and would give meaningless solutions for the wavelet. The mean value of the reflectivity is zero $\langle r_c \rangle = 0$, which results in the desirable zero-mean property of a wavelet, but at the cost of having a wavelet that oscillates indefinitely around zero, thus not fulfilling compact support. Regularization is used to get around it, which penalizes high-norm solutions or solutions with infinitely many non-zero terms. This is the undetermined part of the problem. Namely, Tikhonov regularization serves as a tool to get valid solutions: wavelets with energy concentrated at the beginning.

Another big chapter in this project is the simulation of seismic data using SPECFEM. This becomes our source of real data under the scenario of horizontally placed layers. It is an overly simple scenario that gives us a first well-to-tie approach, with a middle shot surrounded by receivers. This means that the middle gather (or zero-offset trace) in the CMP section collects the vertical activity, while the surrounding characteristics are captured by the adjacent stations.

Stacking yields the final 1D trace that encompasses all the effects that come about when a real wave propagates in a layered media. It helps reduce the noise and improve the quality of the stacked trace. Therefore, the final 1D seismogram contains non-linear phenomena not explained by a convolutional point of view. The convolutional model boils every process down to a 1D convolution of the wavelet with the reflectivity series that only cares about normal reflections. However, we see that for small non-linear perturbations the real stacked data and the convolutional model agree to an acceptable degree.

Regarding NMO correction, we got to work in a range of distances and velocities under which our own implementation of NMO correction performs well, e.g., Figure 5.6. Conversely, this approach relies on the assumption of small offsets, or, alternatively, small changes between offsets. Larger separation among offsets brings about events that cannot be fitted with hyperbolas in the case of the multilayer. So, parallelly, I worked on an alternative DTW method to correct for NMO which rather looks into the similarity between adjacent gathers. Even tough it works in such cases, it is computationally expensive as it requires a large number of points.

With velocity models in the range exposed in Figure 5.7a, the convolution gets to explain the data to a great extent. So it is used as forward model for inversion. Non-linear effects manifest in form of multiples or echos, which are internal reflections inside multilayer that

suck energy out of the system. This leads to the problem of attenuation or transmission loss within the multilayer. When these phenomena are not strong, convolution accounts for them with a different wavelet. One that experiences a shift rotation with respect the Ricker wavelet. This was already sensed and mentioned when discussing Figure 6.1. Yet Tikhonov is able to find that solution for the case of the multilayer.

When inversion was applied, further prior information was required that would tell Tikhonov to find solutions that fulfilled compact support. It entered into the measure of the L-norm in form of exponential functions. This way, we saw that Tikhonov is a robust tool even against noise.

As a final step, the velocity changes between layers were scaled up as a way of introducing stronger multiples. When they become comparable to the data, the solution loses the sense of wavelet. It is not possible to find a solution that, maintaining the properties of a wavelet, did not fit echos as well. This is the bottleneck of the convolution approach. So, this is a simple approach that works in near-linear scenarios.

So this is a beautiful example that works around the small set of near-linear problems, and wells that are wrapped in a closely horizontal structure. In real life, this is unfeasible since effects are far-from linear and more complicated.

In the GitHub page (Martínez, 2022) it is possible to find all the code implementations used for the project.

Chapter 9.

Bibliography

Bianco, Evan (2014). "Geophysical tutorial: Well-tie calculus". In: *The Leading Edge* 33.6, pp. 674–677. DOI: 10.1190/tle33060674.1.

Brown, Raymon L., Wendy McElhattan, and Donald J. Santiago (1988). "Wavelet estimation: An interpretive approach". In: *The Leading Edge* 7.12, pp. 16–19. DOI: 10.1190/1.1439470.

Chen, Shuangquan, Song Jin, Xiang-Yang Li, and Wuyang Yang (2018). "Nonstretching normal-moveout correction using a dynamic time warping algorithm". In: *GEOPHYSICS* 83.1. DOI: 10.1190/geo2016-0673.1.

Fichtner, Andreas (2013). *Full Seismic Waveform Modelling and Inversion*. Springer Berlin.

Foufoula-Georgiou, Efi and Praveen Kumar (1994). "Wavelet Analysis in Geophysics: An Introduction". In: *Wavelets in geophysics*. Academic Press.

GeoSci.xyz. *Seismic travel times*. URL: https://gpg.geosci.xyz/content/seismic/traveltimes.html.

Geuzaine, Christophe and Jean-François Remacle (2002). *Gmsh API*. URL: https://gmsh.info/.

Komatitsch, Dimitri (2018). *User manual SPECFEM2D*. URL: https://specfem2d.readthedocs.io/en/latest/.

Li, Ming and Yimin Zhao (2014). "Seismic Inversion Techniques". In: *Geophysical Exploration Technology: Applications in lithological and Stratigraphic Reservoirs*. Elsevier, pp. 133–198.

Martínez, Pedro (2022). *MSc thesis code*. URL: https://github.com/pedro-mrtnz/msc_thesis.

Menke, William (2012). *Geophysical Data Analysis: Discrete Inverse theory*. Elsevier, Academic Press.

Mosegaard, Klaus (2012). "The Seismic Reflection Method". In: *Lecture Notes 2012*.

Mosegaard, Klaus (2020). "Inverse Problems: an Introduction". In: *Lecture Notes 2020*.

Russell, Brian H. (2009). *Introduction to Seismic Inversion Methods*. Society of Exploration Geophysicists.

Tarantola, Albert (1984). "Inversion of seismic reflection data in the acoustic approximation". In: *GEOPHYSICS* 49.8, pp. 1259–1266. DOI: 10.1190/1.1441754.

Tarantola, Albert (2005). *Inverse Problem Theory and Methods for Model Paramenter Estimation*. Society for Industrial and Applied Mathematics.

Tavenard, Romain. *An introduction to Dynamic Time Warping*. URL: https://rtavenar.github.io/blog/dtw.html#dynamic-time-warping.

Appendix A.

Time-depth conversion

Transform model quantities from a scale of depth (which is the one that provides a picture of the structure) to a scale of time (which is the domain in which the seismic data is acquired) is a recurrent process throughout the project. Therefore, a relationship between depth and time must be established.

In real life, when working with borehole data, it is common to have the sonic logs, among other measurements, in the depth domain. It can be used to establish this relationship between time and depth. The sonic log is the inverse of the velocity, so the integral over a depth interval yield a time-depth relationship (TD):

$$\mathrm{TD} = \int_0^z s(z')\mathrm{d}z' \tag{A.1}$$

Once the time-depth relationship has been set out, it is usually confirmed by generating a synthetic seismogram via convolution and compare it with the real one (inherently in the time domain). If the synthetic seismogram is a good match to the seismic means that the time-depth relationship is robust.

In our tie-to-well approach, we create our own velocity model, which means that we know the value of the velocity at each point of the grid in the depth domain. So, given the precision $\mathrm{d}z$, the simplest way of associating a time t_i to z_i is:

$$t_i = t_{i-1} + \frac{\mathrm{d}z}{v_{i-1}} \quad \text{with } z_i = z_{i-1} + \mathrm{d}z \tag{A.2}$$

where each time step t_i can be computed as time it takes for the wave to go through $\mathrm{d}z$ with velocity v_{i-1}.

Ideally we would like to sample the model with the same time precision as the seismic data. This is, $\mathrm{d}t$. Algorithm 2 shows how this is implemented for the project. Two times are defined t_z and t_t. The former one is defined just like in (A.2), i.e., with sampling rate that depends on $\mathrm{d}z$ and the velocity. On the other hand, t_t is defined according to $\mathrm{d}t$: $t_{i+1} = t_i + \mathrm{d}t$. The number of time steps in t_i is then $t_z/\mathrm{d}t$. Let us imagine now that we would like to convert the density $\rho(z)$ from depth domain to time domain. Through equation (A.2), it is straightforward the one-to-one mapping to $\rho(t_z)$. Then an interpolating function (`interpolation` in the pseudo-code) that takes into account both t_z and t_t is used to obtain the desired quantity $\rho(t_t)$ with sampling rate $1/\mathrm{d}t$.

Algorithm 2: Time-depth relationship. It is written in a pythonic way, with indexes starting at 0. The time t_z is sampled according to $\mathrm{d}z$ and the velocity, whereas t_t follows the given time precision $\mathrm{d}t$. N_z (N_t) is the number of points in the depth (time) domain. The final model in the time domain $q(t)$ is obtained using an interpolating function that takes into account $q(z)$, t_z and t_t.

Input: Velocity model $v_p(z)$, quantity we want to convert $\mathbf{q}_z$, time precision $\mathrm{d}t$ and depth precision $\mathrm{d}z$.

Output: A model quantity in the time domain $\mathbf{q}_t$

$\mathbf{t}_z \leftarrow \mathbf{0}_{N_z \times 1}$
$t_z^0 \leftarrow \mathrm{d}z / v_p^0$
for $i = 1, \ldots, N_z - 1$ **do**
  $t_z^i \leftarrow t_z^{i-1} + \mathrm{d}z / v_p^i$

$N_t \leftarrow \lceil t_z^{N_z - 1} / \mathrm{d}t \rceil$
$\mathbf{t}_t \leftarrow \mathbf{0}_{N_t \times 1}$
for $j = 1, \ldots, N_t - 1$ **do**
  $t_t^i \leftarrow t_t^{i-1} + \mathrm{d}t$

$\mathbf{q}_t \leftarrow \texttt{interpolation}(\mathbf{q}_z, \mathbf{t}_z, \mathbf{t}_t)$

Appendix B.

Dynamic Time Warping

Dynamic time warping (DTW) is a technique used in time series analysis for measuring similarity between two temporal sequences. An idea that was extrapolated to seismic sections for the first time by (Chen *et al.*, 2018) as a workaround of the problem of stretching in normal-moveout correction. In the end, any data that varies with time can be turned into a linear sequence amenable for DTW to analyze.

In general, DTW is a method that calculates an optimal match between two given sequences by means of *alignment-based metrics*, which rely on temporal alignment of the series in order to assess their similarity. Before getting into further details, we will refer to Figure B.1, which compares it with the usual Euclidean metric (not aligning-based). In both cases, the similarity metric is the sum of distances between matched points. We can see that DTW matches distinctive points in both series candidates to be the most similar. It must be stressed the fact that both series in Figure B.1 are shifted vertically for the sake of visualization, but one should keep in mind the zero-shift situation.

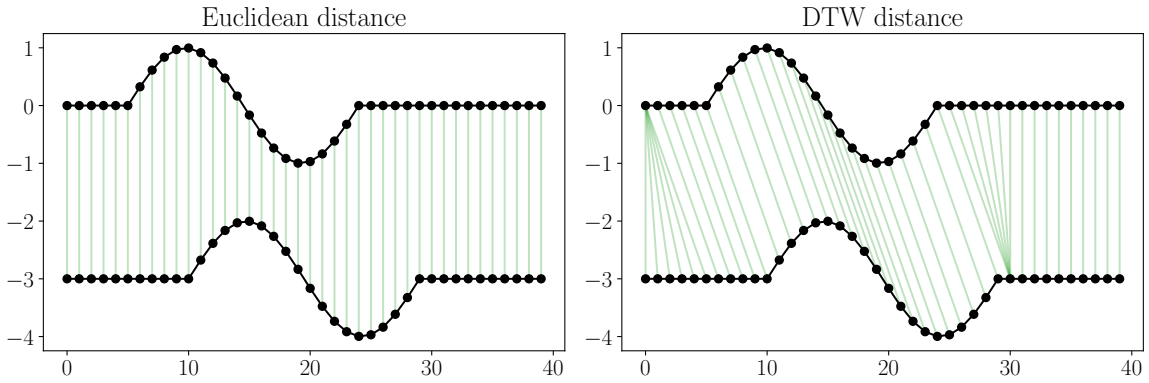


Figure B.1.: Illustration of both DTW (right) and Euclidean (left) metrics. The axes have no units for the sake of simplicity, as it is a mere illustration without real physical meaning. Also, the sequences are shifted vertically, but one should imagine that y-axis values match.

We review now how dynamic time warping works, (Tavenard, n.d.). Let us consider two time series x and x' of respective lengths n and m. Here, all elements x_i and x'_j are assumed to lie in the same p-dimensional space, and the exact timestamps at which the observations occur are disregarded: only their ordering matters.

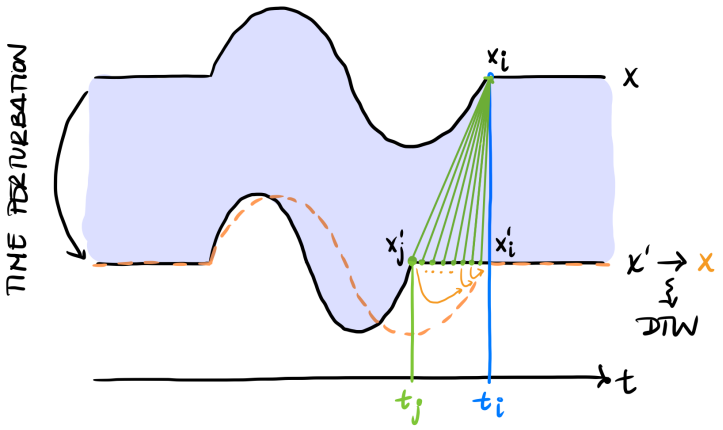


Figure B.2.: Sketch of how time alignment attempts to minimize the Euclidean distance. There are two time series x and x', where a time scale perturbation that shrinks x is applied to obtain x'. The observations $\{x'_j, x'_{j+1}, \ldots, x'_i\}$ (say there are N of them) between t_j and t_i have been all aligned (matched) with x_i, so that when one maps out $\{x'_j, x'_{j+1}, \ldots, x'_i\} \to \{x'_i\}^N$ retrieves the unperturbed x.

Dynamic time warping seeks for the temporal alignment, i.e., matching between time indexes of the two time series. This temporal alignment is constructed so that minimizes the Euclidean distance between aligned series. Figure B.2 illustrates this idea. It exhibits two time sequences x and x', where x' is obtained by applying a time scale perturbation to x that shrinks it. When both series are analyzed using DTW, the algorithm finds out that the data points $\left\{x'_j, x'_{j+1}, \ldots, x'_i\right\}$ (say there are N of them) between timestamps t_j and t_i are aligned (matched) with x_i, since for all the alignments applies that the Euclidean distance is minimum. Therefore, if we map $\left\{x'_j, x'_{j+1}, \ldots, x'_i\right\} \to \{x'_i\}^N$ we retrieve the original x.

But, how does the algorithm work? Formally, the optimization problem can be written as (Tavenard, n.d.):

$$\mathrm{DTW}_q\left(x, x'\right) = \min_{\pi \in \mathcal{A}(x,x')} \left(\sum_{(i,j)\in\pi} d\left(x, x'\right)^q \right)^{\frac{1}{q}} \tag{B.1}$$

Here, an alignment path π of length K is a sequence of K index pairs $\left(\left(i_0, j_0\right), \ldots, \left(i_{K-1}, j_{K-1}\right)\right)$ and $\mathcal{A}\left(x, x'\right)$ is the set of all admissible paths. In order to be considered admissible, a path should meet the following constraints:

1. The first (last) index from x must be matched with the first (last) index from x' (but does not have to be its only match):
   - $\pi_0 = (0, 0)$
   - $\pi_{K-1} = (n-1, m-1)$
2. The sequence is monotonically increasing in both i and j, and all time series indexes should appear at least once:
   - $i_{k-1} \leq i_k \leq i_{k-1} + 1$

- $j_{k-1} \leq j_k \leq j_{k-1} + 1$

Another way to represent a DTW path is to use a binary matrix whose non-zero entries are those corresponding to a matching between time series elements. This representation is related to the index sequence representation used above through:

$$(A_\pi)_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in \pi \\ 0 & \text{otherwise} \end{cases} \tag{B.2}$$

Using matrix notation, dynamic time warping can be written as the minimization of a dot product between matrices:

$$\mathrm{DTW}_q\left(x, x'\right) = \min_{\pi \in \mathcal{A}(x,x')} \left\langle A_\pi, D_q\left(x, x'\right) \right\rangle^{\frac{1}{q}} \tag{B.3}$$

where $D_q(x, x')$ stores distances $d(x_i, x'_j)$ at the power q.

Although the optimization problem in equation (B.1) is a minimization over a finite set, the number of admissible paths (coined Delannoy number) becomes very large even for moderate time series lengths. Assuming m and n are the same order, there exists $\mathcal{O}\left(\frac{(3+2\sqrt{2})^n}{\sqrt{n}}\right)$ different paths in $\mathcal{A}(x, x')$, which makes it intractable to list all paths sequentially in order to compute the minimum.

Fortunately, an exact solution to this optimization problem can be found using dynamic programming. Dynamic programming relies on recurrence, which consists in linking the solution of a given problem to solutions of easier sub-problems. Once the link is known, the dynamic programming approach solves the original problem by recursively solving the sub-problems and storing their solutions for later use, so as not to re-compute sub-problems several times.

In the case of DTW, we rely on the quantity:

$$R_{i,j} = \mathrm{DTW}_q(x_{\to i}, x'_{\to j})^q \tag{B.4}$$

where the notation $x_{\to i}$ denotes times series x observed up to timestamp i (included). Then we can break down the original problem into:

$$\begin{aligned} R_{i,j} &= \min_{\pi \in \mathcal{A}(x_{\to i}, x'_{\to j})} \sum_{(k,l) \in \pi} d(x_k, x'_l)^q \\ &\overset{*}{=} d(x_i, x'_j)^q + \min_{\pi \in \mathcal{A}(x_{\to i}, x'_{\to j})} \sum_{(k,l) \in \pi[:-1]} d(x_k, x'_l)^q \\ &\overset{**}{=} d(x_i, x'_j)^q + \min\left(R_{i-1,j}, R_{i,j-1}, R_{i-1,j-1}\right) \end{aligned} \tag{B.5}$$

where

- $(*)$ comes from constraint 1 on admissible paths π: the last element from an admissible path should match the last elements of the series.
- $(**)$ comes from the contiguity constraint 2 on admissible paths π. A path that would align time series $x_{\to i}$ and $x'_{\to j}$ necessarily encapsulates either:

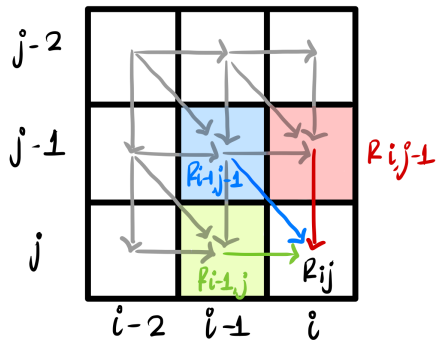


Figure B.3.: Valid DTW transitions.

- a path that would align time series $x_{\to i-1}$ and $x'_{\to j}$, or
- a path that would align time series $x_{\to i}$ and $x'_{\to j-1}$, or
- a path that would align time series $x_{\to i-1}$ and $x'_{\to j-1}$

Figure B.3 illustrates the foregoing reasoning in terms of DTW transitions. The iteration determined by (i, j) can be broken down into the minimum path that leads up to any of the three previous π-points $((i-1, j), (i-1, j-1), (i, j-1))$ and the distance between the current observations x_i and x'_j.

This implies that filling a matrix which would store $R_{i,j}$ is sufficient to retrieve $\mathrm{DTW}_q(x, x')$ as ${R_{n-1,m-1}}^{1/q}$. Algorithm 3 shows how DTW can be implemented.

Algorithm 3: DTW algorithm.

Input: Time series x and x', and norm type q
Output: Optimal path $R_{n-1,m-1}$

$\mathbf{R} \leftarrow \mathbf{0}_{n \times m}$ `// R matrix of size (n x m) filled with zero values`
for $i = 0, \ldots, n-1$ **do**
  for $j = 0, \ldots, m-1$ **do**
    $R_{i,j} \leftarrow d(x_i, x'_j)^q$
    if $i > 0$ ***and*** $j = 0$ **then**
      $R_{i,j} \leftarrow R_{i,j} + R_{i-1,j}$
    else if $i = 0$ ***and*** $j > 0$ **then**
      $R_{i,j} \leftarrow R_{i,j} + R_{i,j-1}$
    else if $i > 0$ ***and*** $j > 0$ **then**
      $R_{i,j} \leftarrow R_{i,j} + \min\left(R_{i-1,j}, R_{i,j-1}, R_{i-1,j-1}\right)$
$\mathrm{DTW}_q(x, x') \leftarrow {R_{n-1,m-1}}^{1/q}$

So far, the most general version of dynamic time warping has been discussed. However, additional constraints can be set, as (Tavenard, n.d.) describes in-depth. Namely, the Sakoe-Chiba band and the Itakura parallelogram constraints are introduced. They typically translate into enforcing nonzero entries in A_π to stay close to the diagonal. However, the were not needed for our purposes.