SEISMIC DEBLENDING: USING ITERATIVE AND COMPRESSIVE SENSING METHODS TO QUANTIFY BLENDING NOISE IMPACT ON 4D PROJECTS

by

Max M. Velasques

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Geophysics).

Golden, Colorado

Date _____

Signed: _____
Max M. Velasques

Signed: _____
Dr. James Simmons
Thesis Advisor

Golden, Colorado

Date _____

Signed: _____
Dr. Paul Sava
Professor and Head
Department of Geophysics

ABSTRACT

Deepwater seismic acquisitions are well-known for their intensive use of financial resources, and the oil & gas industry continuously seeks for ways to acquire seismic data at lower costs to make large offshore projects viable.

Blended seismic acquisitions (also known as *simultaneous acquisitions*) provide an affordable way to acquire seismic data. Blended seismic data makes deblending techniques an essential step in the seismic processing workflow because these methods allow for separate blended shot records, ensuring a high signal-to-noise ratio. Most applications of blended acquisitions rely on conventional 3D projects for exploration purposes, where the residual noise due to seismic interference is not a crucial problem when generating the final seismic image. However, for reservoir characterization, some uncertainties need to be analyzed when time-lapse blended seismic data is considered.

In this project, I implement and compare two inversion-based deblending techniques for use with synthetic and field data: iterative deblending and deblending by compressive sensing. I also quantitatively compare these methods using 4D synthetic data to understand how blending noise and deblending procedures can influence the 4D data.

I found that the tested deblending methods yield very similar results, although their need for computational resources is very different. The analysis with synthetic data before and after the introduction of random noise made the simulation more realistic and allowed me to understand how random noise marginally affects the results. Tests with field data confirmed the effectiveness of the deblending methods, even when missing traces and lateral amplitude variations were present.

From a 4D perspective, under the modeled conditions, my results indicated that the NRMS magnitude (a measurement of repeatability) would be increased by less than 0.5%. This would be the size of the impact of a blended acquisition on our time-lapse project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

Blending operator . . . . . $\mathbf{T}$

Pseudo-deblending operator . . . . . $\mathbf{T}^{+}$

Data vector in the source-receiver-time domain . . . . . $\mathbf{x}$

Blended data vector in the source-receiver-time domain . . . . . $\mathbf{b}$

Data vector in a sparse domain . . . . . $\mathbf{f}$

Number of shots in a sail line . . . . . N_s

Number of receivers in one shot . . . . . N_r

Number of time samples in one trace . . . . . N_t

Total number os samples in a sail line . . . . . N

LIST OF ABBREVIATIONS

Narrow-Azimuth seismic acquisition . . . . . NAZ

Multi-Azimuth seismic acquisition . . . . . MAZ

Wide-Azimuth seismic acquisition . . . . . WAZ

Ocean Bottom Seismic . . . . . OBS

Ocean Bottom Cable . . . . . OBC

Ocean Bottom Nodes . . . . . OBN

Permanent Reservoir Monitoring . . . . . PRM

Signal-to-noise ratio . . . . . SNR

Root Mean Square Amplitude . . . . . RMS

Normalized Root Mean Square Amplitude . . . . . NRMS

Common-midpoint . . . . . CMP

Compressive Sensing . . . . . CS

Source Density Ratio . . . . . SDR

Survey Time Ratio . . . . . STR

Median Filter . . . . . MF

Time-Frequency Denoise . . . . . TFDN

Basis-pursuit denoising . . . . . BPDN

Restricted Isometry Property . . . . . RIP

Reverse Time Migration . . . . . RTM

Amplitude versus Offset . . . . . AVO

ACKNOWLEDGMENTS

First, I would like to express my deepest gratitude to Petrobras for providing me financial support. After 12 years of work, having the opportunity to pursue a master's degree is truly phenomenal.

I would also like to thank my thesis advisor Dr. James Simmons, and Dr. Ali Tura, for all the support and discussions. Their help was essential in turning that set of algorithms and figures into a well-structured project.

I would also like to thank Dr. Ilya Tsvankin, Dr. Paul Sava, and Dr. Eduardo Filpo for accepting to be members of my thesis committee. Their validation of my work was gratifying.

I would like to express my gratitude to Dr. Michael Wakin and Anna Titova for the helpful discussions about compressive sensing. Their suggestions were essential in the development of the project.

I would like to thank my new and old friends who somehow were present in the most difficult moments, especially Odette Aragão, for technical discussions and leisure activities.

In addition, I would like to thank my mother, Vera Velasques, who traveled more than 4,000 miles to help me taking care of my daughter. Without her, this project would not be possible.

Finally, I wish to thank my wife, Andrea Damasceno, for technical discussions and emotional support, and our daughter Beatriz for distracting me precisely when I needed it. This accomplishment would not have been possible without them.

CHAPTER 1

INTRODUCTION

1.1 Background

The costs associated with oil production are noticeably high, mainly when oil reserves are located offshore, where operational challenges for data acquisition and production itself involve the use of equipment adapted to the marine environment.

Among the geophysical methods employed by the industry, seismic is the primary tool for data collection, since it allows companies to quickly obtain data with broad area coverage and good vertical resolution. However, the acquisition of marine seismic data involves major operational challenges as ships, helicopters, and hundreds of technicians are often required. Moreover, its execution may last from a few months to over a year, and its cost is directly related to the time required to acquire data.

During the production phase of an oilfield, the acquisition of new seismic data plays a crucial role in field monitoring, since it allows indirect measurement of various field properties. In this scenario, it is not uncommon for major oil companies to develop permanent reservoir monitoring (PRM) systems for field monitoring. These systems consist of permanently installing receivers on the ocean floor combined with repeated seismic acquisition over a period of time (called 4D[1] seismic acquisition). It allows the receivers to gather valuable information about the field at a specific point in its life. This information is essential in planning future actions that help increase the productivity of the field.

This research, carried out in the Reservoir Characterization Project (RCP), as part of the Petrobras Offshore Project, made use of a PRM seismic data and well logs from the Jubarte field, provided by Petróleo Brasileiro S.A. - Petrobras. The main objective of the project is to analyze the effects of blended data in seismic acquisitions designed for reservoir monitoring.

[1]also known as time-lapse seismic

1.2 Motivation

Although seismic data is of extreme importance in the production phase of an oil field, its use is often limited by the high costs of acquisition, especially in offshore fields.

As discussed by Berkhout (2008), two key parameters of any acquisition are the shooting interval and the recording time, that is, the distance between each detonation of the seismic source and the time receivers are actually recording the signal after each detonation. Seismic acquisitions are designed to ensure that the interval between each shot is large enough to accommodate the desired recording time and prevent any kind of interference between shot records. Otherwise, the energy from interference between shots becomes an additional source of contamination of the seismic signal, and its attenuation, along with several other effects, should be removed or mitigated in the seismic data processing step.

For marine seismic acquisitions, the operational cost is in general related to the time required to acquire the data, which is associated with the vessel speed. Since the vessel speed and the shooting interval are directly related, dropping the non-interference assumption allows an increase in the acquisition speed, and thus, decreases acquisition costs.

Today this approach, known as Simultaneous or Blended Acquisition, is relatively common and, as discussed by several authors (Beasley *et al.*, 1998; Berkhout, 2008; Kumar *et al.*, 2015b; Mahdad *et al.*, 2011), the quality of the final interpretable data is directly related to how separable the blended shots are. The process of separating seismic records recorded during blended acquisitions is called seismic deblending.

From an exploration perspective, the available deblending techniques are very effective in recovering independent shot records suitable for imaging applications. However, for reservoir characterization purposes, some uncertainties need to be analyzed when time-lapse blended seismic data is considered, because any additional noise can compromise the quality of the 4D signal, making it impracticable to use the data for such purpose.

I propose to quantitatively analyze the impact of acquiring blended time-lapse seismic data for reservoir monitoring. With that in mind, I will extract and analyze the 4D seismic

response (4D difference) from seismic data that has gone through a deblending workflow in order to understand if lower-cost blended time-lapse acquisitions are adequate for reservoir monitoring.

1.3 Objective

The main objective of this thesis is to analyze the impact of the acquisition of marine blended time-lapse seismic data for reservoir monitoring. Since blended acquisitions generally make it necessary to apply deblending techniques, a secondary objective is to test different deblending methods to verify their effectiveness.

The main steps followed to fulfill these goals are summarized below:

1. Simulate a time-lapse OBS acquisition by generating two synthetic acoustic datasets (baseline and monitor), whose velocity and density models are based on information extracted from a Brazilian offshore production field;

2. Implement, apply, and compare the effectiveness of two deblending techniques (iterative deblending and deblending by compressive sensing) on the modeled synthetic data;

3. Using the synthetic dataset, analyze how blending noise[2] affects the 4D signal;

4. Check the effectiveness of the implemented deblending techniques in an OBS dataset acquired over an offshore field in Brazil.

1.4 Marine Seismic Acquisition Overview

The main objective of any seismic acquisition is to adequately illuminate targets in the subsurface (usually oil and gas reservoirs). For this purpose, seismic sources and receivers generate and record seismic waves, respectively (Vermeer, 2002).

Air guns are the primary seismic source for marine acquisitions, and in general, they are grouped in arrays that aim to correctly target seismic energy and eliminate some types of

[2]noise due to the interference of seismic records. The blending noise is, in fact, recoverable signal if the appropriated method is applied

noise[3]. The depth of the source (actually the array of air guns) plays an essential role in the acquired seismic data amplitude spectrum (ghost effect due to the source depth). The first notch in the source ghost amplitude spectrum is controlled by the source depth, where the shallower the source, the higher the maximum usable frequency (as long as the source is still submerged). Caldwell & Dragoset (2000) give a useful overview of seismic air gun arrays geometry, and Laws *et al.* (2008) provide some insights about possible issues for time-lapse acquisitions.

From the receiver side, several technologies meet specific needs. The most common techniques, from the cheapest to the most expensive, are:

- **Streamers** — cables that are towed by the sea surface carrying and connecting receivers to the recording vessel;
- **Ocean Bottom Cables (OBC)** — temporary underwater cables carrying and connecting receivers to the recording vessel;
- **Ocean Bottom Nodes (OBN)** — autonomous receivers temporarily installed on the ocean floor. After recording the seismic data, these nodes need to be recovered, have their contents downloaded and have their batteries recharged before moving to a new position;
- **Permanent Reservoir Monitoring (PRM) systems** — receivers permanently installed on the ocean floor, in general, connected by cables to production stations that continuously record active and passive[4] seismic data.

Streamer acquisition is still by far the most common seismic acquisition technology in the industry. The most common geometry is the acquisition of narrow azimuth (NAZ) using simultaneously 12 cables or fewer (Duey, 1996). However, technologies such as wide azimuth acquisition (WAZ) with more than 18 cables (Threadgold *et al.*, 2006) have become more

[3]the most common is the bubble effect
[4]there is no external seismic source, only natural or induced seismicity generates seismic waves

and more frequent, making it possible to reduce the imprints introduced by the acquisition geometry (acquisition footprint) inherent in cable acquisitions.

Similar to what happens with the seismic sources, the depth of the cables impacts the data frequency content (due to the receiver side ghost). The shallower the cables, the higher the maximum usable frequency. Additionally, the depth of the cables affects the noise level. The shallower the cables are positioned, the more susceptible to noise are the data recorded (Vermeer, 2002).

For 4D surveys, where repeatability plays an important role, ocean bottom technologies are more suitable (Duey, 2011). Stationary-receiver techniques (Vermeer, 2002), like OBC and OBN acquisitions, have become common in recent decades, mainly due to the combined use of pressure and velocity receivers[5] (Barr, 1997), and have better resolution, better repeatability for time-lapse acquisition, and the high signal-to-noise ratio. Due to the high operating costs, they are used mainly in reservoir characterization surveys, which usually have smaller areas to be covered and require some level of repeatability.

In general, OBNs can achieve higher levels of repeatability than OBCs, since the receiver (nodes) are carefully positioned by remotely operated underwater vehicles (ROVs). However, because of technological limitations, the nodes are arranged in sparse grids that need to be compensated by high density shot grids (Vermeer, 2002). A common limitation with this technology is related to coupling issues that need to be addressed during the positioning of receivers (Olofsson *et al.*, 2007). Besides that, since the nodes are self-contained receiving and recording units, this means they are not connected to vessels or platforms, so they need to use batteries that need to be recharged after some period of time.

One particular type of ocean-bottom seismic acquisition is the permanent reservoir monitoring (PRM) system. As the name suggests, it is a system composed of receivers that are permanently placed on the ocean floor, and they are usually connected to the surface (on a production platform) by cables. Combined with high-density shot grids, PRM systems are

[5]3D/4C receivers

the ultimate acquisition technology for reservoir monitoring/characterization, and generate the best results in terms of time-lapse seismic (Thedy *et al.*, 2013). From the available technologies, it has the highest initial cost for implantation. However, after that, new seismic acquisitions over the field become only an exercise of repeating the same shot grid over time. In 2015, 13 PRM systems were active, including Valhall's, running since 2003 (Caldwell *et al.*, 2015). Figure 1.1 shows the geometry of a PRM system installed on an offshore production field in Brazil (Dariva *et al.*, 2016).

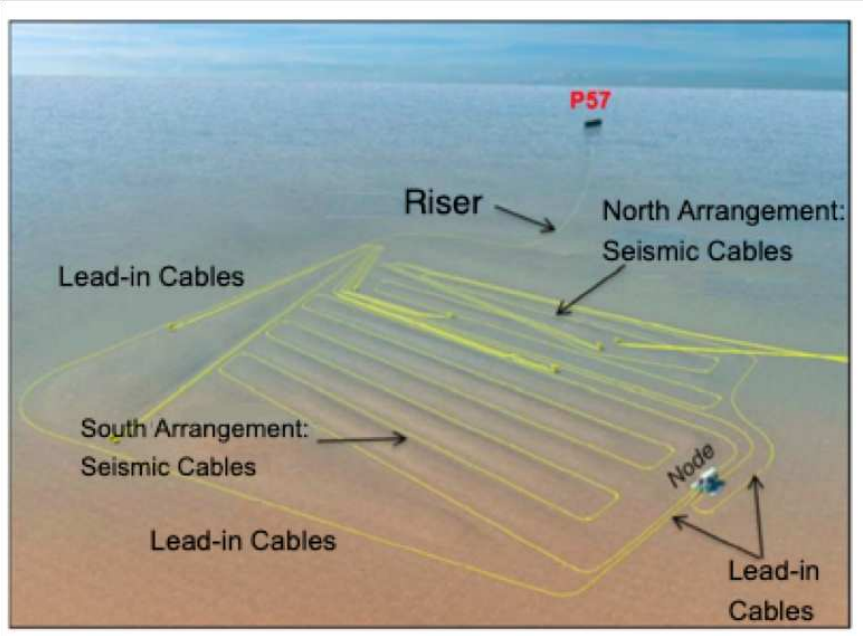

Figure 1.1: PRM system installed on an offshore production field in Brazil. Adapted from Dariva *et al.* (2016).

Regardless of the technology, time-lapse acquisitions consist of repeat seismic surveys to capture changes in the reservoir (even if small), due to the field production. The first survey of a time-lapse project is called the baseline, and later surveys are called monitor, followed by the number of repetitions (for example, monitor 1, monitor 2, etc.).

As mentioned earlier, repeatability is essential in time-lapse projects because production effects are captured from the difference between monitor and baseline surveys. The most common way to measure repeatability is using the normalized RMS (NRMS) of the dif-

ferences (Kragh & Christie, 2002), measured outside the reservoir, where changes are not expected. Lower NRMS values mean higher repeatability.

1.5 Traditional vs Blended Seismic Acquisition

In the previous section, I described the main recording technologies used in marine acquisitions in addition to the most common seismic source, air gun arrays. However, as briefly commented, non-interference between shots was assumed, which, together with the recording time, directly affects the productivity and costs of a survey. Since the recording time is related to the target depth in the subsurface, in general, this parameter cannot be changed easily.

In practice, the relationship between the recording time and the shooting interval is given by the source vessel speed. For example, for values widely used in exploration projects (shooting interval of 25 meters and a vessel speed of 5 Knots, approximately 2.5 meters per second), the maximum recording time is 10 seconds. In order to be able to reduce the shooting interval to 12.5 meters (common value for reservoir characterization projects) and keep the recording time to 10 seconds, it is necessary to reduce the vessel's speed in half. Unfortunately, it is not always possible to do this, especially in streamer surveys, where the presence of sea currents needs to be balanced by a higher speed of the ship, in order to maintain the receiver cables in a stable depth.

Figure 1.2 shows a schematic representation of an ocean bottom acquisition with a recording time of 10 seconds and a shooting interval of 25m and 12.5m with their respective speeds. It is possible to observe that, to acquire data with a shooting interval of 12.5 meters, it is necessary to reduce the ship's speed to 2.5 knots, which doubles the time and cost of a seismic acquisition.

As discussed by Berkhout (2008), blended acquisitions are designed considering interference between shot records. That means they allow to keep appropriate recording times and shooting intervals without the need to decrease the ship's speed (which in general increases costs). Figure 1.3 shows a schematic representation of a seismic acquisition with sequen-

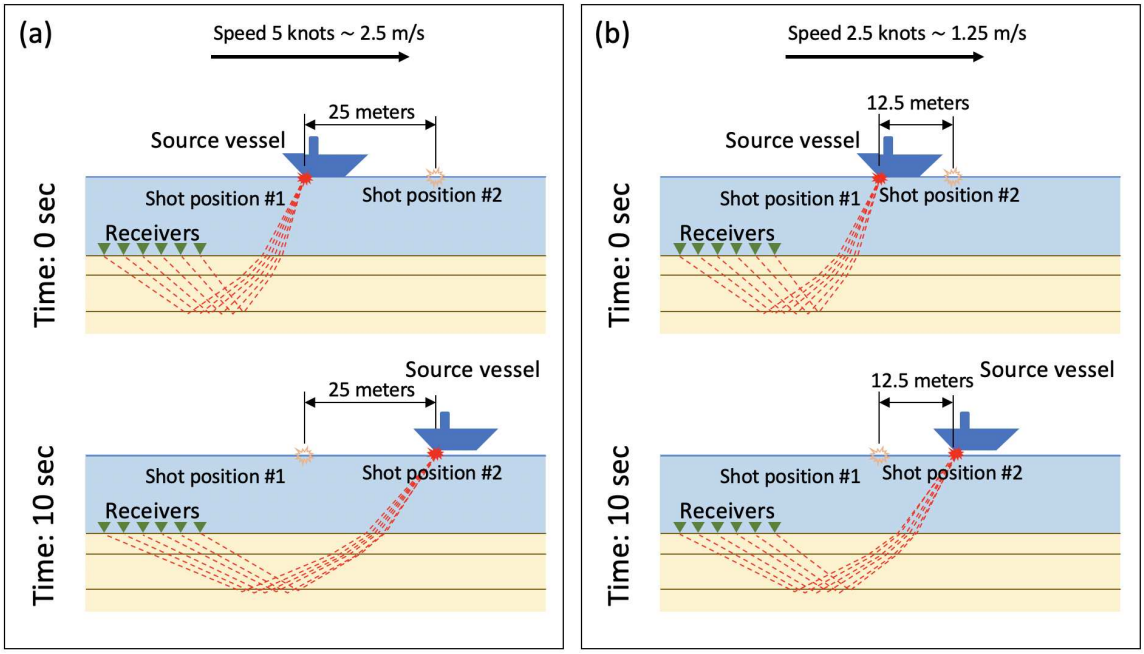

Figure 1.2: Schematic representation of ocean bottom acquisition with a recording time of 10 seconds and shooting interval of 25 meters (a) and 12.5 meters (b).

tially blended shots and random firing times. This blending procedure is called random time-dithering scheme (Wason, 2017).

In order to allow the proper separation of shot records during the seismic processing stage, the signature of the sources must be encoded[6] (Mahdad *et al.*, 2011). In marine acquisitions, the most common encoding method is using random firing times, because when the data is rearranged in domains other than the common-shot domain, part of the blended shots appear as random noise, allowing the application of deblending techniques. For this encoding method, the exact firing times need to be recorded, as this information is crucial for all deblending techniques. Figure 1.4 shows an unblended[7] OBC shot gather and a numerically blended shot gather considering six blended sources. The irregular moveout observed in the data is due to the geometry of the receiver cables (Figure 1.1) not being linear.

[6]Method used to simplify source separation. For example, variable firing times for marine surveys, or variable vibratory energy for land surveys

[7]by definition, a conventional seismic record acquired without interference

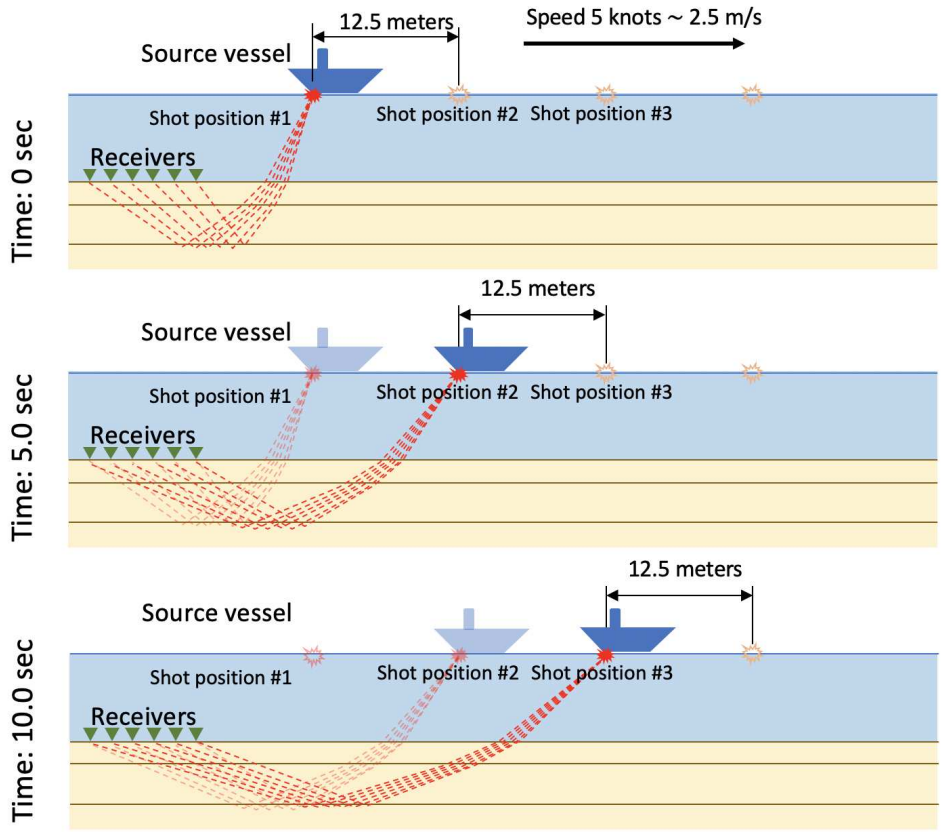

Figure 1.3: Schematic representation of a blended acquisition with a recording time of 10 seconds and shooting interval of 12.5 meters considering an overlap between two sequential shots.

In practice, due to variations in sea current, waves, or even due to the surface weather, the speed of the source vessel can vary from shot-point to shot-point. Although small, these almost random variations introduce fluctuations in the time interval between shots on the order of hundreds of milliseconds (Aaron *et al.*, 2016), and as discussed before, these variations are essential[8] for the application of deblending methods.

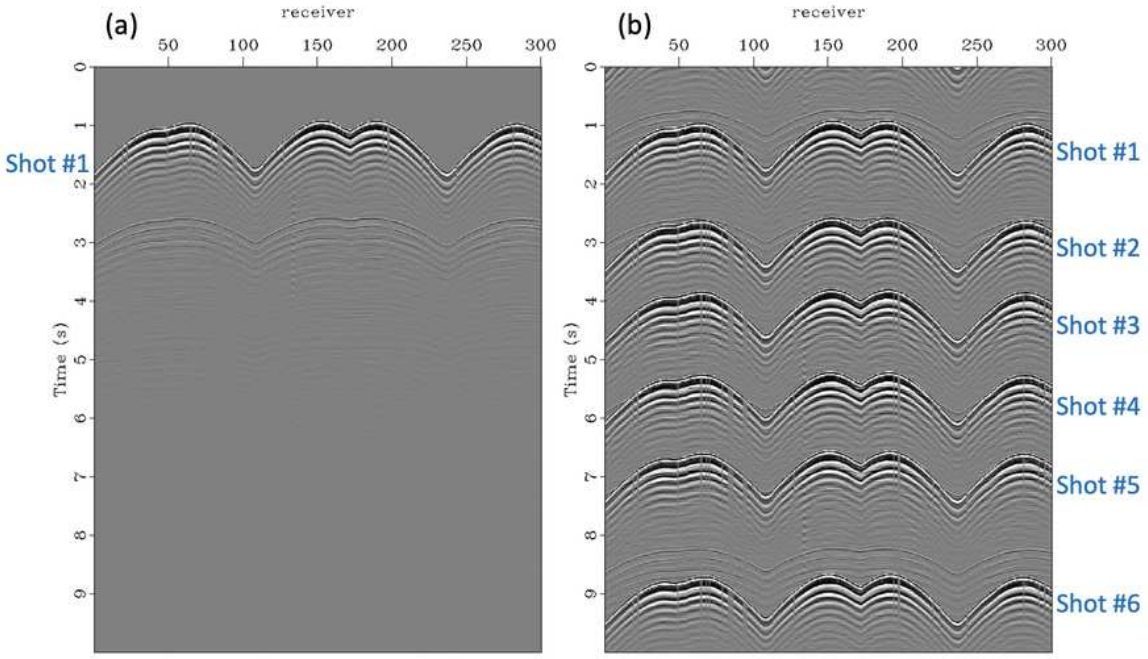

Figure 1.4: Comparison between an unblended shot gather extracted from a real OBC acquisition (a), and a numerically blended shot gather with 6 shots (b).

Blended geometries with more than two simultaneous shots, or even with more source vessels, are frequently used by the industry in exploratory campaigns. Survey design for blended seismic acquisitions was discussed by Blacquière *et al.* (2009).

1.6 Blended Seismic Acquisition Performance Indicators

During the design phase of a blended acquisition, it is essential to be able to measure the performance or productivity that will be achieved, when compared to a traditional

[8]See Chapter 2 for more details on how randomness affects the blended seismic data

acquisition. To that end, Berkhout (2008) introduced two key indicators: the source density ratio and the survey time ratio.

The source density ratio (SDR) compares the number of sources that would be used in a blended acquisition with that of a traditional acquisition:

$$SDR = \frac{\text{number of sources in the blended survey}}{\text{number of sources in the unblended survey}} \tag{1.1}$$

The survey time ratio (STR) compares the number of days it would take to acquire the data using the traditional unblended approach with the number of days for a blended acquisition:

$$STR = \frac{\text{number of acquisition days in the unblended survey}}{\text{number of acquisition days in the blended survey}} \tag{1.2}$$

STR is of great importance for marine acquisitions where costs are directly associated with the time of the survey.

Combining these indicators is possible to measure total economic performance as:

$$\text{Blended Acquisition performance} = SDR \times STR \tag{1.3}$$

For SDR and STR, the higher the values, the more efficient and economical the seismic acquisition will be.

1.7 Seismic Deblending Techniques

The step after the acquisition of any seismic data is the seismic processing. For blended data, this is not different. In fact, seismic processing for blended data is inherently the same as for conventional data, with one exception, the need to generate individual seismic shot gathers (deblending).

Seismic deblending is one of the first processing steps, aiming to separate blended shot gathers so the data can be treated as a conventional seismic. The main deblending methods available in the literature are filtering based (where well-known denoise workflows are applied), or inversion based (in general, seeking sparse solutions).

Filtering methods treat the blended data as random in some specific domain, so that random noise attenuation tools can be applied to remove the blending noise. Chen *et al.* (2014a) describe a deblending procedure where they take advantage of the incoherent property of blending noise using normal moveout (NMO) and median filtering in the common-midpoint domain (CMP) to remove the blending noise and recover deblended data. Gan *et al.* (2016) use a structural-oriented median filter (MF), following the structural direction of seismic profiles, to attenuate the blending noise. Several other methods are applicable, as long as they allow one to discriminate signal and random noise in some domain.

Filter based deblending is simple and effective, however, due to the need for the signal and noise to be separable, they do not always produce the desired results, and attenuation of the signal is very common during the blending noise removal. Because of this, inversion deblending methods are known to generate superior results (Chen *et al.*, 2018, 2013; Huang *et al.*, 2018; Kumar *et al.*, 2015b; Mahdad *et al.*, 2011; Peng & Meng, 2016).

Inversion based deblending methods use matrix representations of the seismic data, where an objective function is built and solved through established inversion techniques (see Chapter 3). Mahdad *et al.* (2011) apply the least-squares solution combined with denoise techniques to model the blending noise and subtract it from the blended data. Kumar *et al.* (2015b) propose to apply two compressive sensing approaches (sparsity promotion and rank minimization) to recover deblended data. Cheng & Sacchi (2015) described a deblending technique by separating and reconstructing blended data via iterative rank reduction.

In addition to the methods already described, several authors directly use imaging and inversion techniques, instead of trying to separate blended shots. Chen *et al.* (2015) propose to migrate the blended data without deblending, treating the imaging step as a least-squares inversion problem.

In this thesis, only the deblending methods by inversion will be described, highlighting the: iterative deblending and the deblending by compressive sensing (CS).

CHAPTER 2

BLENDED MARINE SEISMIC ACQUISITION

A proper mathematical description of a problem is just as important as the solution itself. This holds true for seismic deblending. In this chapter, I detail the methodology used to describe seismic data in matrix form.

Since in this thesis, different deblending methods are analyzed, a description in the time-space domain is preferable, as will be made clear in Chapter 3. However, it is worth to mention that the description, in the frequency domain, introduced by Berkhout (1982), is also applicable. Despite the domain, the representation of blended seismic data as matrices is essential to the mathematical description of the seismic deblending methods.

2.1 Matrix Representation

Regardless of the recording technology employed, a marine seismic acquisition can, in general, be subdivided into sail lines[9] (usually hundreds of them) with shot positions and intervals that have been previously defined[10]. Using this subdivision, one way to describe a marine acquisition is by sail line. Figure 2.1 shows a post-plot map, with more than 400 sail lines, 25 meters apart, acquired on an OBS acquisition offshore Brazil (Barros, 2015).

Considering a sail line as a set of N_s shot records, where for each shot, N_r receivers were active, and each receiver recorded the signal for a time t (with N_t samples), this can be represented as a 3D cube, as illustrated by Figure 2.2. Slices in this cube allow us to extract records in different domains. For example, a horizontal slice $\mathbf{A}$ contains all shots and receivers for a specific time as indicated by the horizontal green line shown on the top face of the cube; a common shot record $\mathbf{B}$ contains all time samples and receivers; and a common receiver gather $\mathbf{C}$ contains all time samples and shots.

[9]Valid for 3D and 4D acquisitions. For 2D, as there is only one acquisition line, this subdivision is not necessary

[10]See Vermeer (2002) for more information about seismic acquisition design

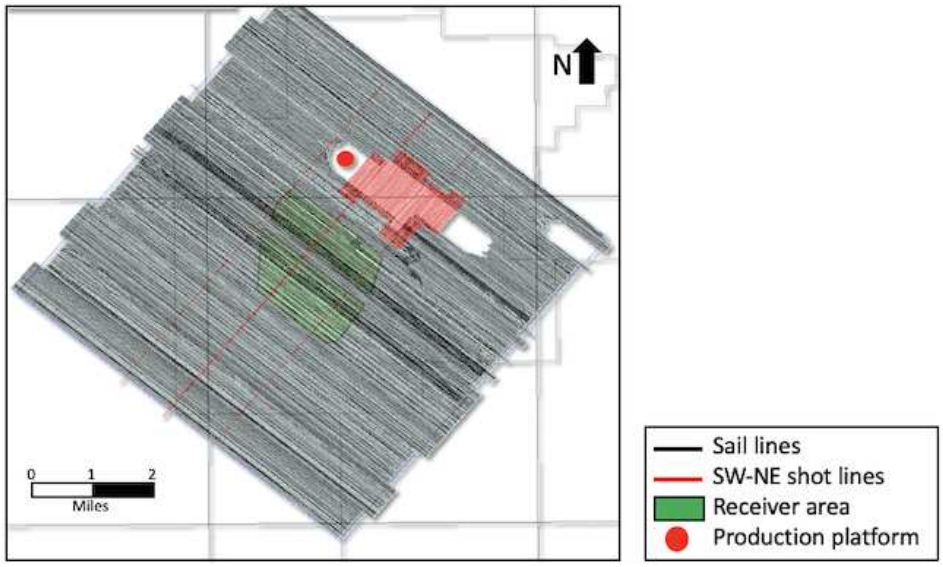

Figure 2.1: Post-plot of an OBS acquisition with its sail lines in black, receiver lines in green, and an obstruction (production platform) in red. Adapted from Barros (2015).

In the common receiver domain (Figure 2.2, slice C), it is possible to observe that each trace belongs to a different shot, and all shots of that sail line are present. Therefore, blended data can be easily represented in this domain, as will be detailed in Section 2.3.

Considering a common receiver gather as an $N_s \times N_t$ matrix , it is possible to map this 2D record into a column vector $\mathbf{x}_r$ of size $N_s N_t$. When extended to blended acquisitions, the description of shot records with more than one source consists of combining elements of $\mathbf{x}_r$ with some type of *encoding*[11], which further allows the separation of individual shots. Figure 2.3 illustrates the data vectorization process for a hypothetical receiver gather with 8 shots and 10 time samples.

The above representation does not impose restrictions on the spatial regularity of shots and receivers. However, it does consider that each receiver recorded all shots of a sail line. If this assumption is not valid, the vector representation can still be used, as long as the absence of shots in one common receiver gather is taken into account.

[11] operational procedure for blending seismic data that take advantage of mathematical principles that simplify future data separation

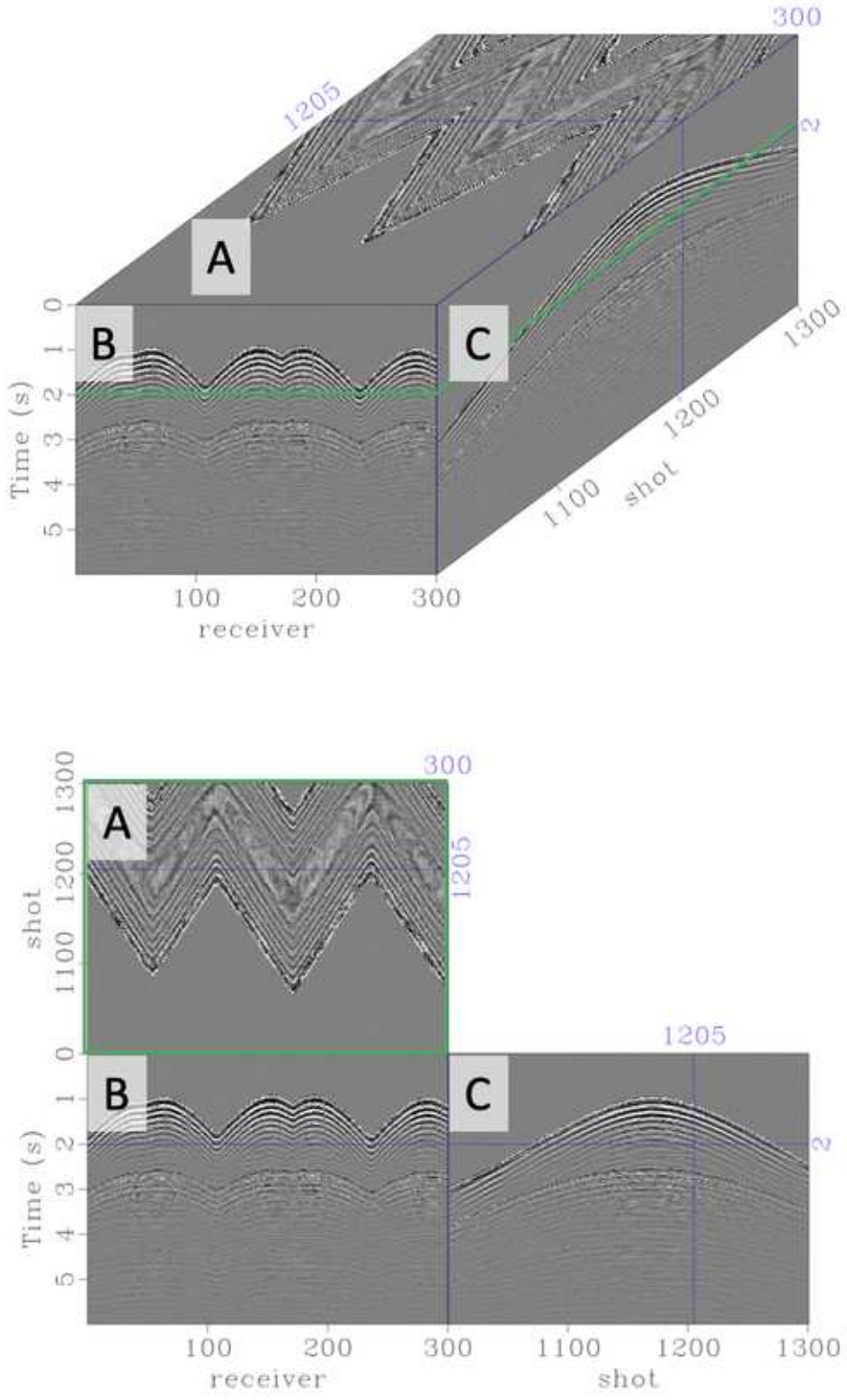

Figure 2.2: At the top, 3D representation of a sail line, in the time-space domain, acquired offshore Brazil. At the bottom, 2D slices extracted from the cube faces **A**, **B** and **C**.

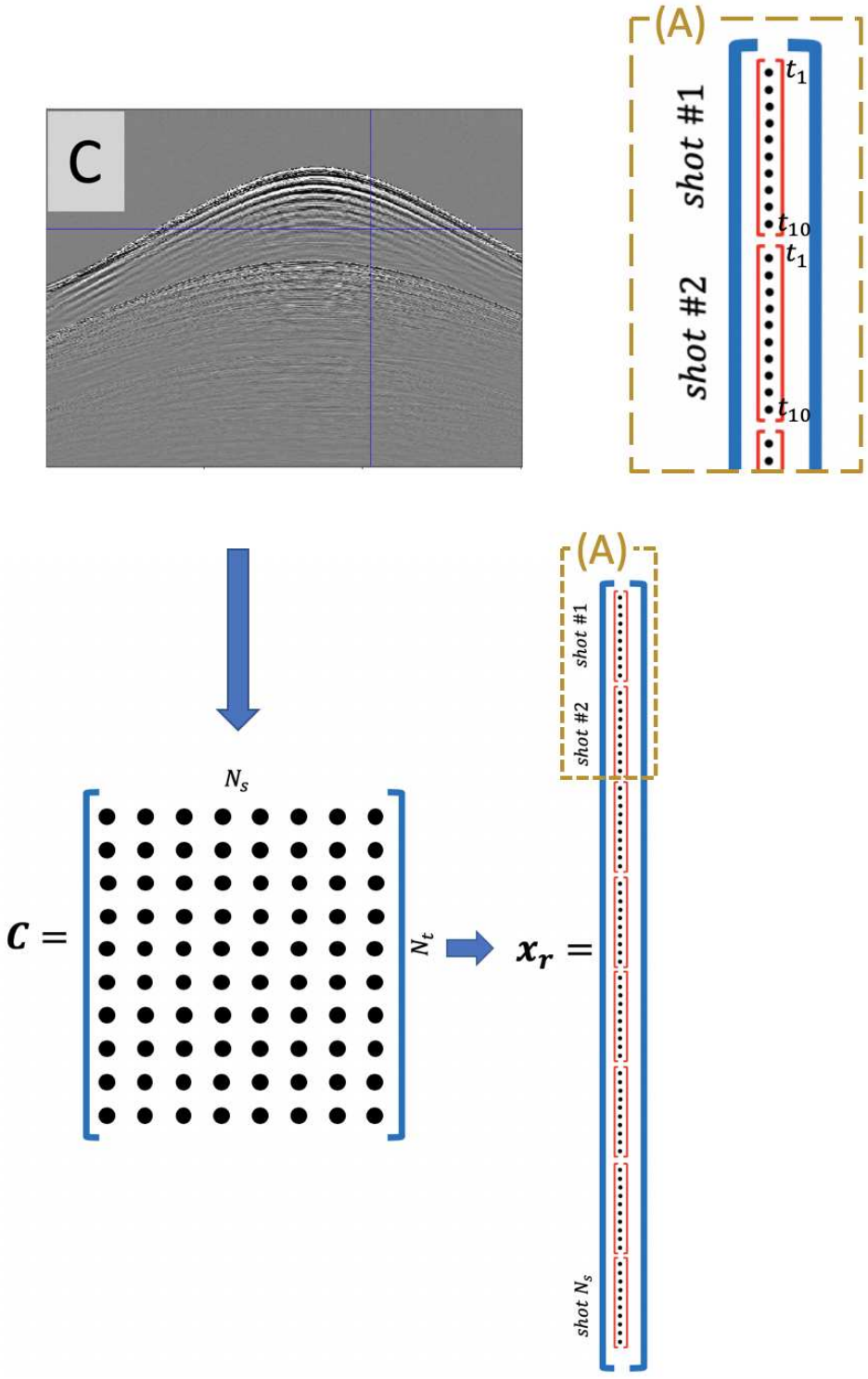

Figure 2.3: Pictorial illustration of the vectorization process on a receiver gather with $N_s = 8$ and $N_t = 10$. At the top right, a zoom of region **A**, illustrating some elements of $\mathbf{x}_r$ with the first and last time sample of each shot.

2.2 Random Time-dithering Scheme

Before discussing how blended data can be described mathematically, it is crucial to understand the effects of regular and random firing times on the seismic signal.

As highlighted by Berkhout (2008), blending is a process that yields an incoherent source wavefield, which means a simple combination of source wavefields is not enough to generate a blended seismic data properly.

For a seismic acquisition, coherent wavefields are generated, when the time interval between the detonation of each source is regular. An extreme case is when all sources are detonated simultaneously. Figure 2.4a shows the resulting wavefield when simultaneously detonated point sources combine or interfere with each other and merge into one continuous wavefront[12]. In this example, a plane wavefront is generated (coherent).

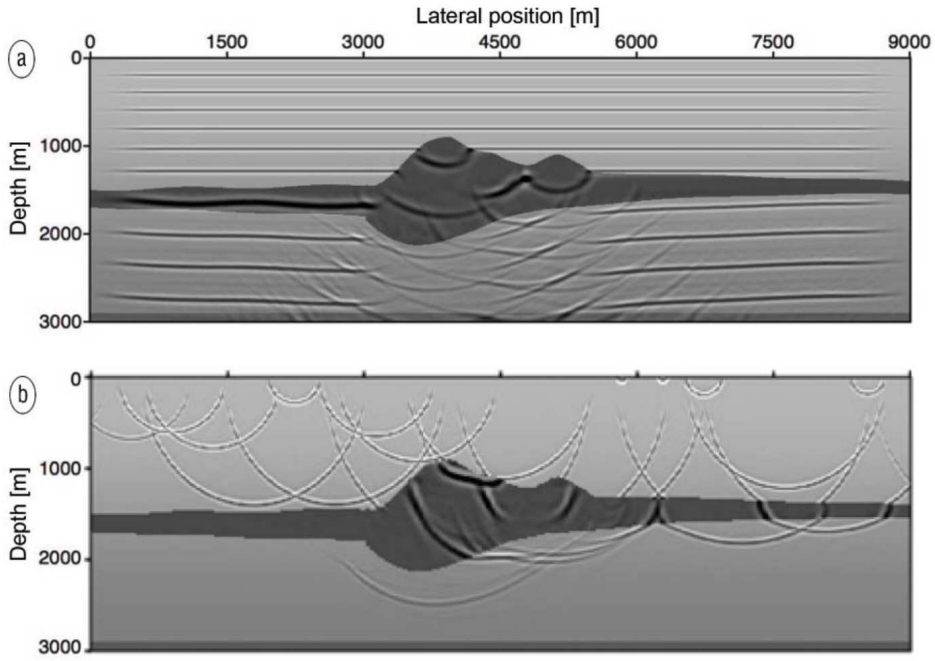

Figure 2.4: Synthesis of point sources generating a plane source wavefield, which means a coherent source wavefield. (a). Simulation of a blended source wavefield, which means an incoherent source wavefield (b). Extracted from Berkhout (2008).

[12]Berkhout (1982) calls this process of synthesis

In contrast, Figure 2.4b shows the same sources from (a), but now with random initiation times, generating separate wavefronts (incoherent). The extraction of individual shot gathers, in a combined configuration of single sources, is only possible when the generated wavefronts do not merge into a unique wavefront. This condition preserves the full temporal and spatial bandwidth, which is essential for seismic deblending.

Figure 2.5 illustrates how different arrangements of the sources (plots b to d) affect the seismic response in the t-x and f-k domains when compared to the response of a single shot (plot a). In all simulations, 20 sources were considered (regularly spaced by 50 meters). This analysis is similar to that performed by Mahdad (2012).

In the first configuration of combined sources (Figure 2.5b), all sources are fired simultaneously. This generates approximately a horizontal plane wave traveling vertically, which due to reduced number of dips in t-x domain appears as equally spaced vertical stripes in the f-k domain, consistent with the source spacing. The coherence of the generated wavefront affects the bandwidth of the recorded data since several wavenumbers are not present in the data. The second configuration (Figure 2.5c) shows sources detonating with a regular firing time interval of 0.3 seconds(regular time-dithering). This generates regular dipping wavefronts, which can be seen as dipping stripes in the f-k domain. As before, this regular configuration eliminates several wavenumbers from the data. The last simulation (Figure 2.5d) considers sources detonating with a random firing time interval of 0.3 ± 0.15 seconds (random time-dithering). This random configuration generates a more homogeneous f-k spectrum, which means that, all wavenumbers are preserved, making seismic deblending possible.

If a more realistic scenario, when non-regular shot spacing is considered, the transformation from the irregular t-x domain to the f-k domain introduces noise into the data, making the f-k spectrum more complex.

If the data are analyzed in domains other than the shot domain, it is possible to observe the impact of a random configuration on the data. Figure 2.6 compares a shot gather, a receiver gather, and a CMP gather to analyze the seismic response of blended data.

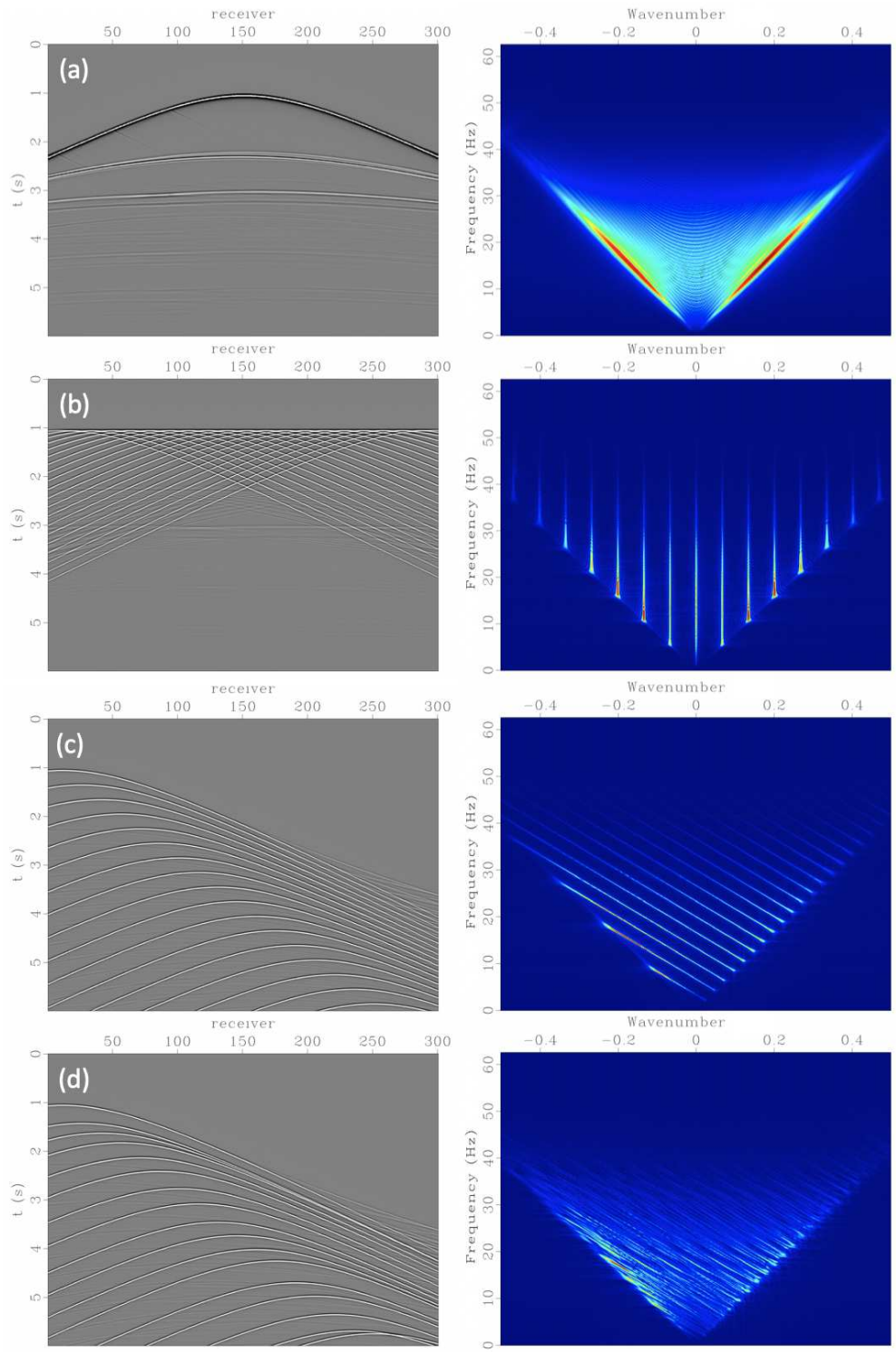

Figure 2.5: Shot records and f-k spectra of different arrangements of sources. Single unblended shot (a). Sources fired simultaneously (b). Sources with a regular firing time interval, which means a regular time-dithering scheme (c). Sources with a random firing time interval (d).

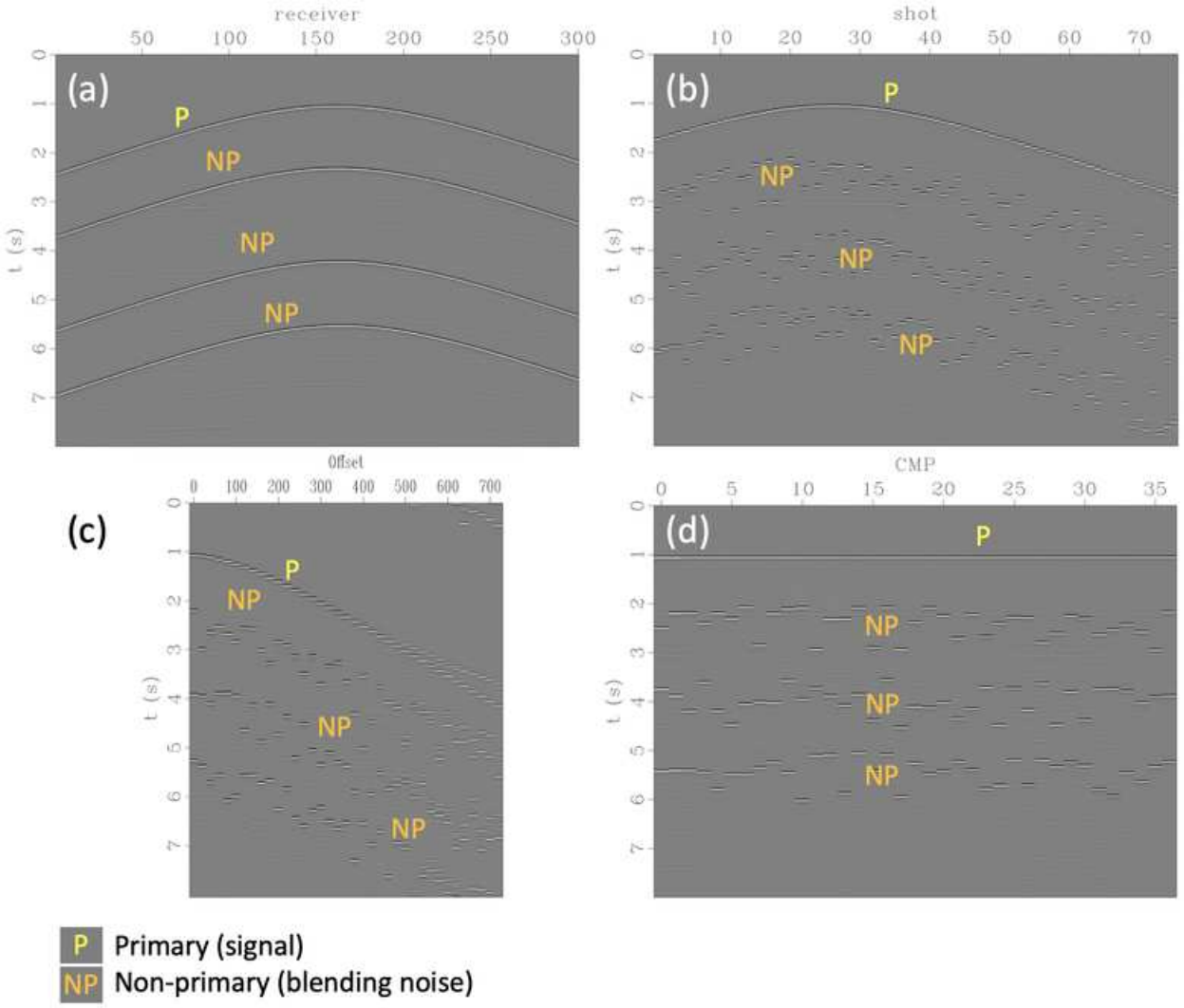

Figure 2.6: Blending noise in different domains. Common shot domain (a), common receiver domain (b), common mid-point domain (c), and common offset domain (d).

Figure 2.6a illustrates a shot gather with reflections from different sources appearing as coherent events. Because of this coherence, the application of deblending techniques in this domain is more challenging. However, when the data is sorted into common receiver gather, common CMP or common offset gathers (Figure 2.6b, c, d, respectively), the events from non-primary sources[13] appear as random noise.

Deblending techniques by filtering, or even by inversion that use some kind of coherence analysis exploit the fact that blended events appear as random noise in some domains, to generate better results. In general, the common receiver domain is the most used for deblending applications.

Considering the need for some sort of randomness between the firing times of the shots, a question that arises is: How random should these times be? Aaron *et al.* (2016) demonstrated with a field experiment that for a constant shooting interval survey, the natural shot-to-shot timing variations caused by natural vessel speed changes between shot locations were adequate for the deblending. In their experiment, an average absolute shot-to-shot variation in firing timing around 120ms was enough for deblending applications. Because the objective of my research is consistent with their observations, I will use the random time-dithering scheme (with small firing time variations) in all simulations and analysis.

2.3 Blending Procedure

As discussed in Section 1.5, the main objective of a blended acquisition is to drop the non-interference assumption between shot records in order to increase productivity. In this thesis, my focus is the blending of sequential shots, applying the random time-dithering scheme, which allows for an increase in the speed of the source vessel (Figure 1.3). However, blended acquisitions can have different designs (Lin & Herrmann, 2009; Vermeer, 2002), including surveys with several source vessels shooting simultaneously. The following description is valid for one or multiple source vessels, as long as the blending operator considers all the seismic sources involved, regardless of their spatial positions, or complexity.

[13]sources detonated at reference times other than zero time.

2.3.1 2D Representation

Starting from the description introduced in Section 2.1, a common receiver gather can be represented as a column vector $\mathbf{x}_r$ of size $N_s N_t$, where segments of $\mathbf{x}_r$ are related to different shots. If we consider that $\mathbf{x}_r$ contains all shots[14], the blending process is based on the application of an operator $\mathbf{T_r}$ on $\mathbf{x}_r$, capable of adequately combine[15] segments of $\mathbf{x}_r$, generating a blended data $\mathbf{b}_r$. Mathematically this process is described by:

$$\mathbf{T_r}\mathbf{x}_r = \mathbf{b}_r, \tag{2.1}$$

where $\mathbf{b}_r$ is the blended receiver gather in the form of a column vector of size n_b, and $\mathbf{T_r}$ is the blending operator, in the time-space domain, of size $n_b \times N_s N_t$.

Equation 2.1 describes a way to combine source wavefields for generating blended seismic data.

As discussed in Section 2.2, random firing times generate the necessary amount of incoherence between source wavefields (Figure 2.5), which means the natural way to *encode*[16] the sources for a marine survey is to use the so-called *linear phase encoding*.

Linear phase encoding is the application of time-shifts between sources that are expected to interfere. This denomination comes from the description of seismic data in the frequency domain, where a time-shift is equivalent to a multiplication by a phase term[17] ($e^{-i\omega t}$).

In the time-space domain, which is the representation I chose, the linear phase encoding is characterized by the inverse Fourier transform of phase terms $e^{-i\omega\tau_{kl}}$, where τ_{kl} is the randomized time-shift between sources k and l. This ends up being delta functions with different time lags ($\delta[t - \tau_{kl}]$) in the time domain.

In practice, the blending operator $\mathbf{T_r}$ can be represented as a matrix. In this case, building a time-space domain blending operator, with the linear phase encoding, means creating a

[14] if this assumption is not valid, a description in 3D needs to be considered

[15] combination as a blending scheme, not as a synthesis process

[16] operational procedure for blending seismic data that take advantage of mathematical principles that simplify future data separation

[17] in the frequency domain, a blending operator, in general called $\mathbf{\Gamma}$, can be written as $\mathbf{\Gamma}_{T_{kl}} = e^{-i\omega\tau_{kl}}$, where τ_{kl} is a randomized time-shift (Mahdad *et al.*, 2011).

matrix of dimensions $n_b \times N_s N_t$ with a few elements set to "1", and the majority of the elements set to "0". Figure 2.7 illustrates the $\mathbf{T_r}$ operator for a hypothetical receiver gather with 8 shots and 10 time samples.

When more than one source vessel is used, blended surveys with 2, 4 or even more simultaneous sources are conducted. However, for surveys where the desire is to increase the speed of the source vessel, continuous collection[18] of the data (as *continuous records*) is more appropriate. This process consists of recording seismic signals as a single long record containing all shots for that sail line. Figure 2.8 illustrates a record with four sources and one continuous record.

For conventional seismic acquisitions where data is acquired continuously, individual shot gathers can be recovered by "cutting" the records in N_s small records with a fixed number of time samples N_t, where N_s is the number of shots. However, for blended seismic acquisitions, seismic deblending techniques are required in order to separate and extract individual shot records.

2.3.2 3D Representation

As highlighted in the previous section, the blending process can be fully described by individual common receiver gathers, as long as the condition that all shots are present in all receivers holds. However, as will become clear in Chapter 3, a 3D description is desired when deblending methods based on sparse solutions are used.

Considering that the operator $\mathbf{T_r}$ is the same for all receivers, equation 2.1 can be extended for the entire sail line using the *Kronecker product*. The Kronecker product is a generalization of the outer product, and it is defined as an operation between two matrices of different sizes.

[18] combined with the random time-dithering scheme

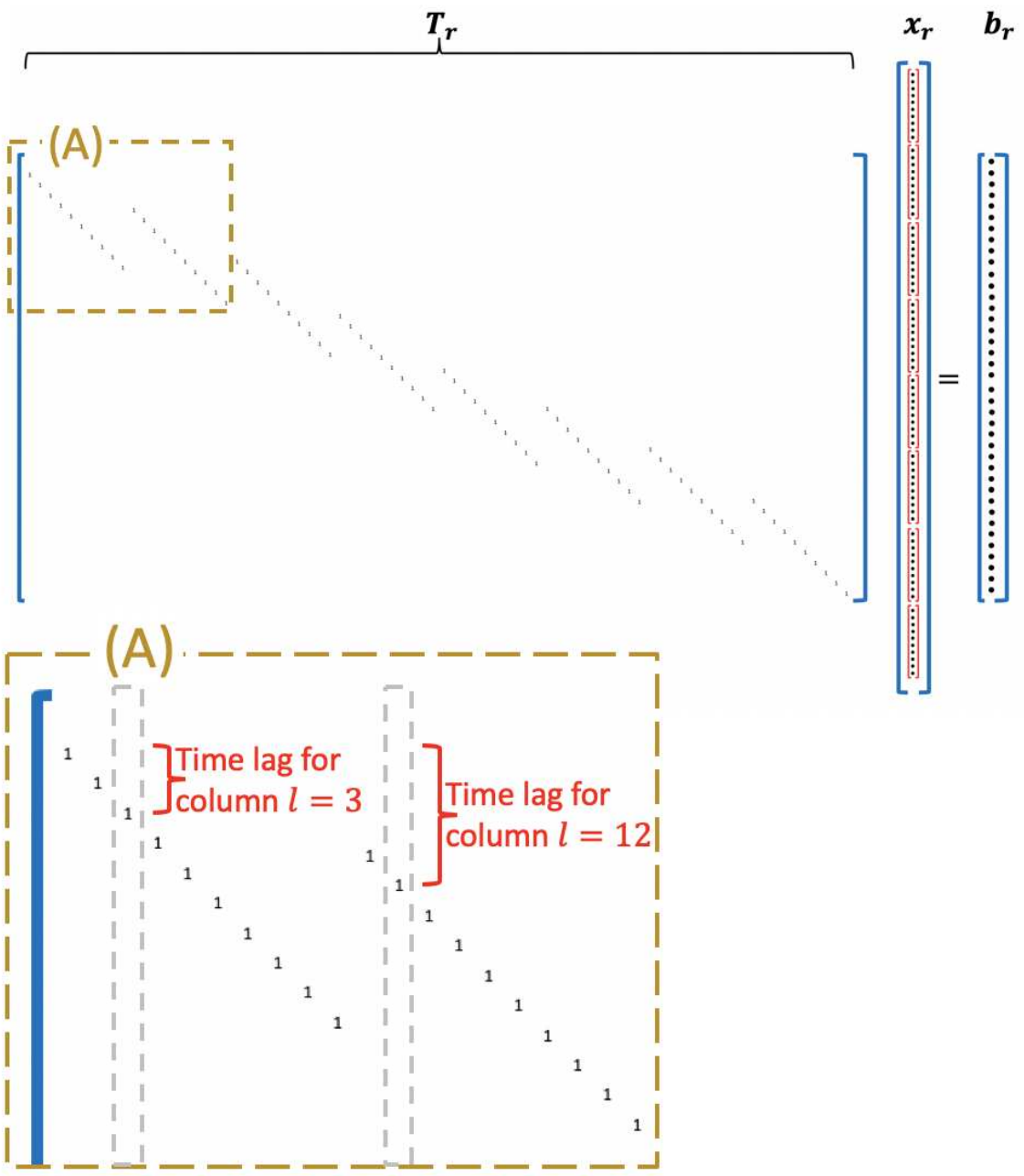

Figure 2.7: At the top, a pictorial illustration of a 2D blending operator ($\mathbf{T_r}$) and its application to an unblended vectorized receiver gather, generating a blended receiver trace. At the bottom, a zoom of region $\mathbf{A}$, illustrating some columns of $\mathbf{T_r}$ and the time lags for the delta functions. The elements of $\mathbf{T_r}$ that were not displayed have a value of zero.

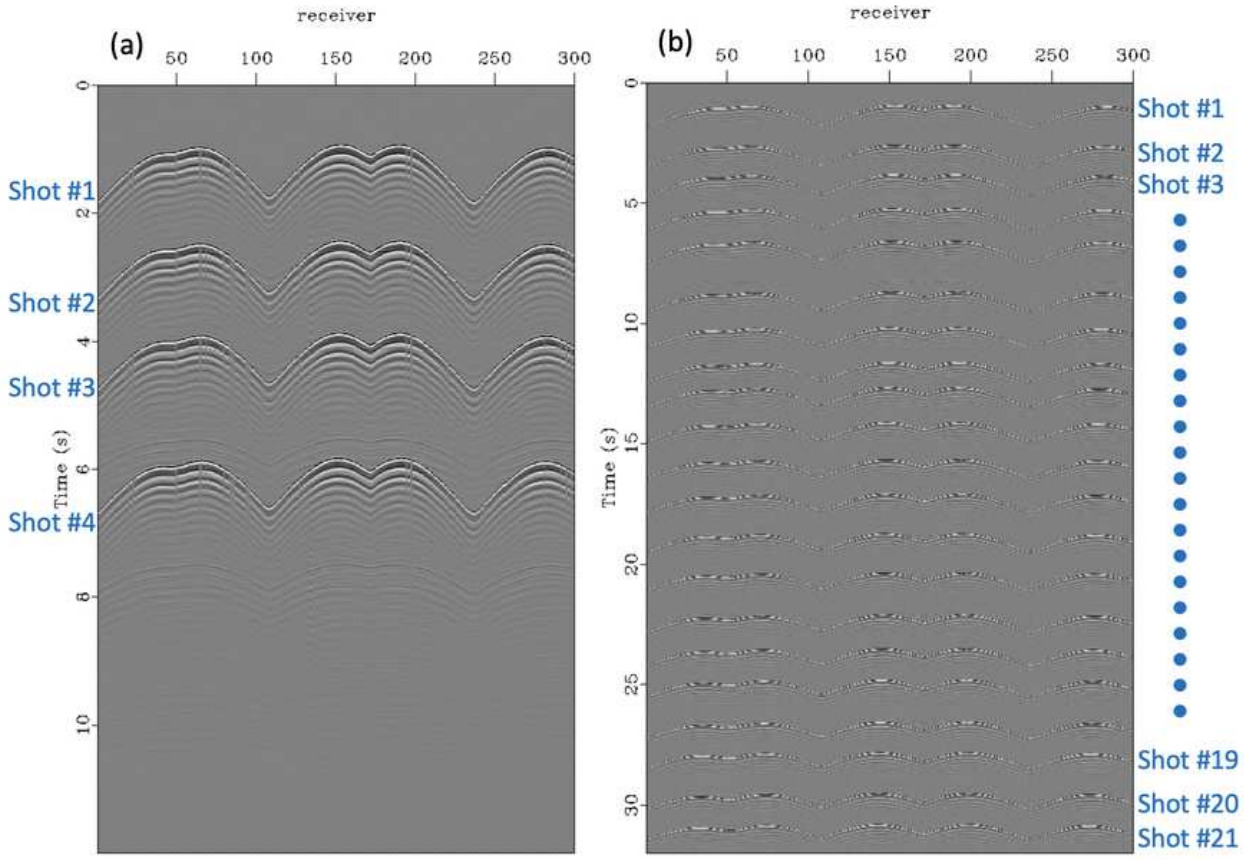

Figure 2.8: Numerically blended shot gather (a) with 4 individual shots detonated at a random time interval of 2.5 ± 0.4 seconds. Numerically blended continuous shot gather (b) with 21 individual shots detonated at a random time interval of 2.5 ± 0.4 seconds.

Equation 2.2 illustrates the Kronecker product between two matrices:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \otimes \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} = \begin{bmatrix} a_{11} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} & a_{12} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \\ a_{21} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} & a_{22} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \end{bmatrix}, \quad (2.2)$$

where the Kronecker product is denoted by the symbol $\otimes$.

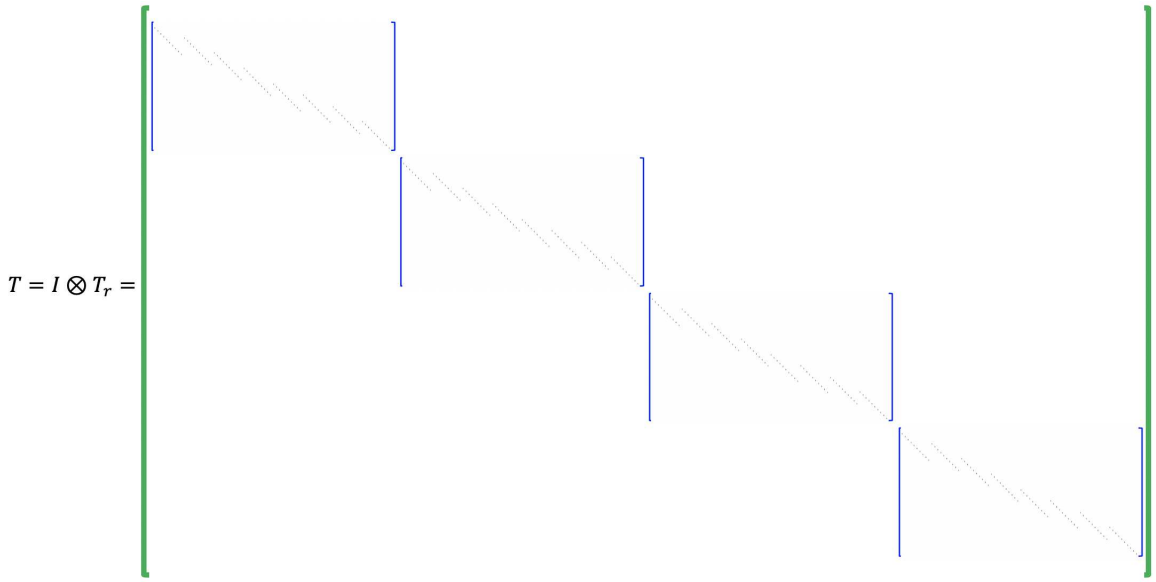

Figure 2.9: Pictorial illustration of a 3D blending operator T. The blue matrix from Figure 2.7 was repeated N_r times in matrix $\mathbf{T}$ along the main diagonal. The elements of $\mathbf{T}$ that are not displayed have a value of zero.

Equation 2.3 gives the 3D blending operator $\mathbf{T}$ in terms of the 2D operator $\mathbf{T_r}$:

$$\mathbf{T} = \mathbf{I} \otimes \mathbf{T_r}, \quad (2.3)$$

where $\mathbf{I}$ is an $N_r \times N_r$ identity matrix, and $\mathbf{T}$ is an $N_r n_b \times N_r N_s N_t$ matrix. Figure 2.9 illustrates the 3D blending operator[19].

[19]Please note that the definition of the 3D operator is based on a 2D matrix. The operator is called 3D, because it acts on the entire 3D data cube at the same time, and not just on 2D slices..

Following the definition given by equation 2.3, the generalization of equation 2.1 is:

$$\mathbf{Tx} = \mathbf{b} \tag{2.4}$$

where $\mathbf{x}$ is the fully vectorized sail line with size $N = N_r N_s N_t$, and $\mathbf{b}$ is the vectorized blended data of size $n = n_b N_r$ for $n << N$.

If the condition that all shots are present in all receivers does not hold, a fully 3D representation needs to be used. In this case, the construction of the blending operator $\mathbf{T}$ becomes dependent of the relation between shots and receivers.

2.4 Blending Procedures Remarks

In this chapter, I introduced the mathematical description of blended seismic data, focusing on the main objective of the thesis, which is to acquire single vessel seismic data faster and at a lower cost.

I used the natural subdivision of marine seismic acquisitions into sail lines to describe the data, which included all shots that may have some overlap with previous or posterior neighbors.

In my description, I demonstrated how to rearrange unblended seismic data in a column vector (Figure 2.3), and how to combine its elements through the application of a blending operator, generating blended seismic data. The matrix description is essential in Chapter 3, where the seismic deblending methods is presented as an inverse problem.

The investigation of how to combine different sources was also presented based on the theory introduced by Berkhout (2008). From this, I demonstrated that a random firing time interval scheme is essential for the separation (deblending) of blended seismic data.

Finally, although the description presented was limited to the time-space domain, it will become clear in Chapter 3 that it is applicable to any transform domain, if an inverse transform is performed on the data before applying the blending operator.

CHAPTER 3

SEISMIC DEBLENDING - THEORETICAL BACKGROUND

Seismic deblending is one of the initial processing steps, intending to separate blended data so it can follow the processing flow as a conventional seismic. In the literature, one can generally find two main categories of deblending methods: filtering based (where well-known denoise workflows are applied), or inversion based (in general, seeking sparse solutions).

As summarized in Section 1.7, filtering methods treat the blended data as random in some specific domain, so that random noise attenuation tools can be applied to remove the blending noise. F-k filtering and structural-oriented median filters (MF) are frequently used. On the other hand, inversion based deblending methods use matrix representations of the seismic data (as the one introduced in Chapter 2), where an objective function is built and solved through established inversion techniques. Because inversion based deblending methods are known to generate superior results (Chen *et al.*, 2018, 2013; Huang *et al.*, 2018; Kumar *et al.*, 2015b; Mahdad *et al.*, 2011; Peng & Meng, 2016), those methods are the main focus of my thesis.

In this chapter, two deblending methods by inversion are presented. The first method uses an iterative approach (Mahdad, 2012) that combines a least-squares solution with coherence filters to eliminate the so-called blending noise and recover unblended gathers (Section 3.4). The second method, based on compressive sensing techniques (Kumar *et al.*, 2015b; Wason, 2017), aims to separate shots solving a sparsity-promoting optimization problem. In both methods, the prior information that the signal to be recovered is sparse becomes an important constraint in the objective function to be optimized (Section 3.5).

3.1 Problem Statement

In Chapter 2, I presented how to mathematically describe blended acquisitions in the time-space domain. Equation 2.4 describes a forward modeling procedure that generates

blended data from unblended records by the application of a linear operator $\mathbf{T}$.

The goal with seismic deblending is to recover unblended records from blended data. For this, it is necessary to perform matrix inversion. Since, in both cases, there are more unknowns than equations, the system of equations is underdetermined. Which means that an inverse problem needs to be formulated.

Inverse problems are the basis of several scientific methods, where observations are used to infer the properties of a system based on mathematical models. Using seismic inversion conventional terminology, for seismic deblending: the "observed data" is the blended data $\mathbf{b}$; the "predicted data" is the deblended data $\mathbf{x}$; and the blending operator $\mathbf{T}$ is the one that connects data and solution spaces, encapsulating the physics of the problem.

The minimum data misfit is the criteria to find optimal mathematical solutions. However, this criterion not always generates valid solutions, which can be overcome, incorporating proper constraints.

3.2 Pseudo-inverse Solution - Pseudo-deblending

The first approach for solving inverse problems is through the least-squares solution, which is formulated by evaluating the best fit for the observed data.

For underdetermined systems, there is an infinite number of solutions, and the most common approach is to find the smallest solution subject to the best fit. Mathematically, we have:

$$\min_{\mathbf{x}} \|\mathbf{x}\|_2^2 \text{ subject to } \mathbf{b} = \mathbf{T}\mathbf{x}, \tag{3.1}$$

which is a constrained optimization problem.

In order to find the solution for equation 3.1, it should be converted into an unconstrained problem using Lagrange multipliers [20]:

$$J(x, \lambda) = \|\mathbf{b} - \mathbf{T}\mathbf{x}\|_2^2 + \lambda \|\mathbf{x}\|_2^2 . \tag{3.2}$$

[20] such that the derivative approach to finding a minimum of an unconstrained equation can be applied (Luenberger, 1997).

where λ is a Lagrange multiplier, and λ and $\mathbf{x}$ are independent variables.

Minimizing equation 3.2 is equivalent to solving equation 3.1. Mathematically, it can be done by taking the derivative of $J(\mathbf{x}, \lambda)$ with respect to its parameters, and setting it to zero. This gives us the least-squares solution for equation 3.1:

$$\mathbf{x} = \mathbf{T}^H(\mathbf{T}\mathbf{T}^H)^{-1}\mathbf{b}, \tag{3.3}$$

where $\mathbf{T}^H$ is the transpose[21] of $\mathbf{T}$ (also known as Hermitian), and $\mathbf{T}^H(\mathbf{T}\mathbf{T}^H)^{-1}$ is the pseudo-inverse of $\mathbf{T}$ (Penrose, 1955), denoted by $\mathbf{T}^+$.

By construction, the matrix form of $\mathbf{T}$ has a valuable property, which is, its rows are mutually orthogonal. This would let us to infer that $\mathbf{T}$ is an orthogonal[22] matrix. However, this is not true, since $\mathbf{T}$ is not a square matrix. In this case, an extension of orthogonality for non-square matrices can be used (Abadir & Magnus, 2005), and $\mathbf{T}$ is labeled as a *semi-orthogonal matrix*[23]. In this case, the following property holds:

$$\mathbf{T}\mathbf{T}^{\mathbf{H}} = \mathbf{I}, \tag{3.4}$$

where $\mathbf{I}$ is the identity matrix. Based on that, the pseudo-inverse matrix can be written as

$$\mathbf{T}^+ = \mathbf{T}^H(\mathbf{T}\mathbf{T}^H)^{-1} = \mathbf{T}^H. \tag{3.5}$$

Consequently, equation 3.3 can be simplified to

$$\mathbf{x} = \mathbf{T}^H\mathbf{b}. \tag{3.6}$$

The solution given by equation 3.6 is called *pseudo-deblending*. Figure 3.1 shows a synthetic unblended data in the common receiver domain, followed by the respective blended data (generated by equation 2.4), and the pseudo-deblended data.

Initially, one can observe the effects of the blending operator on the data in the common receiver domain. Comparing Figure 3.1a and Figure 3.1b, one can see that the number of traces (directly related to the number of shots) in the blended record is less than that of the

[21] if $\mathbf{T}$ is a complex matrix, then $\mathbf{T}^H$ is the conjugate transpose of $\mathbf{T}$

[22] an orthogonal matrix is a square matrix whose columns and rows are orthogonal unit vectors

[23] a semi-orthogonal matrix is a non-square matrix whose columns or rows are orthogonal unit vectors

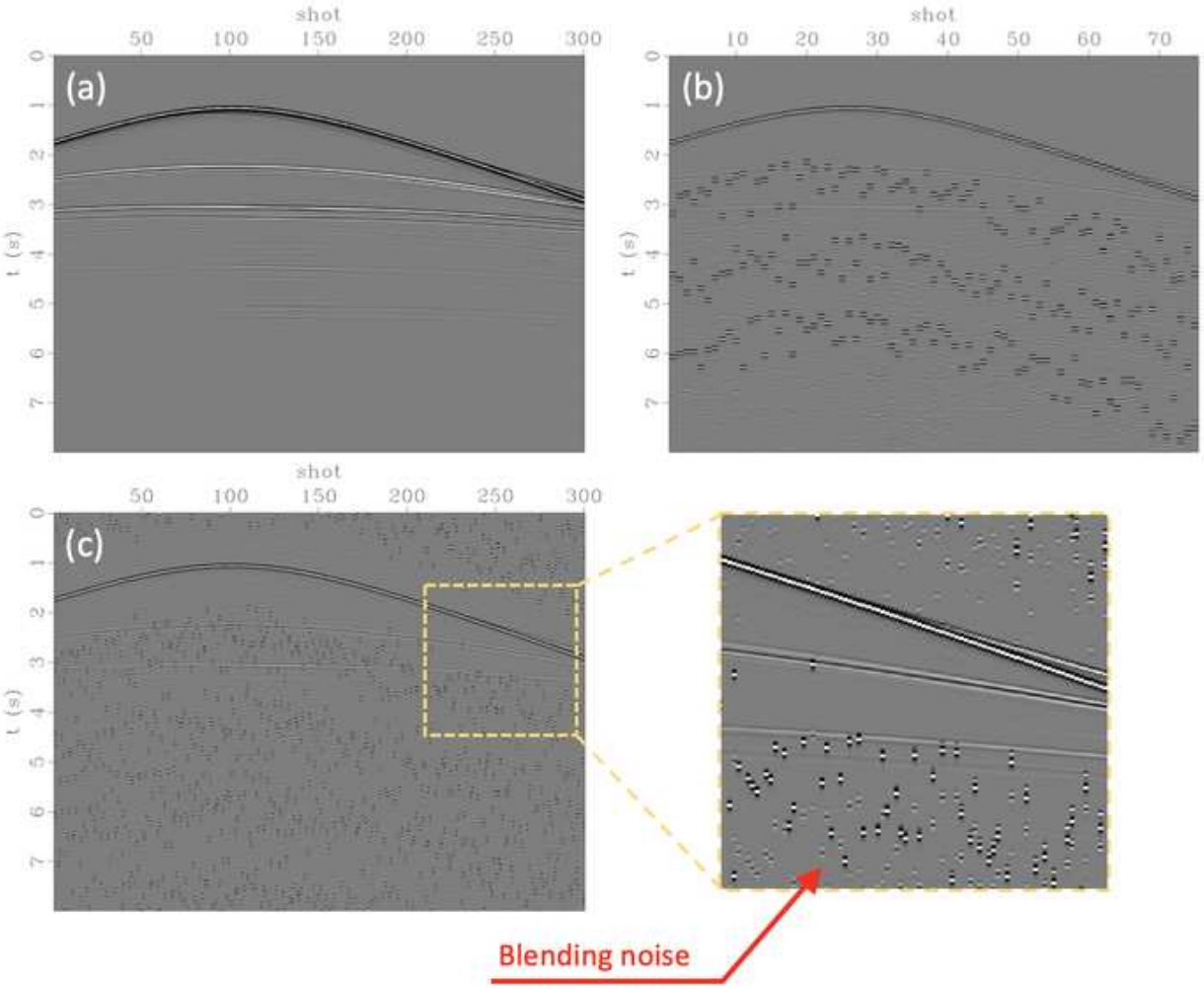

Figure 3.1: Simulation considering 4 blended sources. Unblended receiver gather (a), blended receiver gather (b), and pseudo-deblended receiver gather (c). The blending noise generated by the pseudo-deblending method is highlighted by the red arrow. All plots have amplitude clipping to highlight the noise.

unblended data. Data compression occurs in this domain. Since the simulation considered four blended sources, the number of traces is four times smaller.

Now, analyzing Figure 3.1c, one can observe that the number of traces in the pseudo-deblended receiver gather is the same as in the original. The pseudo-deblending decompresses the data correctly, restoring the initial amount of traces. It also decodes the recovered traces by time-shifting them to the time zero reference. Although this process reconstructs and decodes the data correctly, it is contaminated by a random noise called *blending noise*.

Mathematically, the blending noise is generated because of the non-orthogonality between the columns of $\mathbf{T}$ (Figure 2.7), which prevents the cancellation of cross-terms generated in this process. In practice, the pseudo-deblended data is a valid solution for this underdetermined problem. However, it is clearly not the desired solution, which suggests that extra constraints need to be imposed on our solution.

The amplitude of the *blending noise* is directly related to the type of encoding used. Therefore, for an encoding composed of time-shifts, the amplitude of the noise is equivalent to that of the signal.

3.3 Penalized Least-Squares Problem

The results generated by minimizing the objective function described by equation 3.1 were not satisfactory. Therefore, the introduction of prior knowledge is required to narrow the space of possible solutions.

Seismic data, when represented in some transform domain - for example, Fourier, Wavelet, or Curvelet transform - presents a useful characteristic called *sparsity*. A sparse representation means that most of the signal energy, in the transform domain, is concentrated in a small number of coefficients (Hennenfent & Herrmann, 2006).

Mathematically, the sparsity of a signal can be measured by evaluating the number of non-zero elements necessary to represent it in some transform domain. This can be performed using the ℓ_0-norm of the transformed signal ($\|\mathbf{Sx}\|_0$).

Going back to the definition of the blended and deblended data vectors ($\mathbf{b}$ and $\mathbf{x}$, respectively) in Chapter 2, we know that $\mathbf{b}$ has size n and $\mathbf{x}$ has size N, where $n \ll N$. Considering the solution to our problem (deblended data) being sparse in some domain, only a few elements of $\mathbf{Sx}$ are necessary to reconstruct $\mathbf{x}$. The sparsity k of a vector is calculated by evaluating its ℓ_0-norm ($k = \|\mathbf{Sx}\|_0$). If $\mathbf{Sx}$ is sparse, this means $k \ll N$.

If one takes advantage of the sparsity of $\mathbf{x}$, the objective function from equation 3.1 can be replaced by:

$$\min_{\mathbf{x}} \|\mathbf{Sx}\|_0 \text{ subject to } \mathbf{b} = \mathbf{Tx}, \tag{3.7}$$

where $\mathbf{S}$ is a sparse transform, and $\|\mathbf{Sx}\|_0$ is the number of non-zero elements in $\mathbf{Sx}$.

Finally, using a Lagrange multiplier the constrained problem is equivalent to:

$$J(x, \lambda) = \|\mathbf{b} - \mathbf{Tx}\|_2^2 + \lambda \|\mathbf{Sx}\|_0 \tag{3.8}$$

where λ is a regularization parameter. Minimizing equation 3.8 is equivalent to solving equation 3.7.

Unfortunately, equation 3.8 cannot be solved by traditional minimization methods[24]. Moreover, the ℓ_0-norm is a combinatorial problem, which means that the number of possible solutions increases rapidly with the size of the data. Based on that, alternative procedures for obtaining sparse solutions must be used.

3.4 Iterative Deblending

In this section, the theoretical basis of the first deblending technique analyzed in this thesis, the iterative deblending, will be presented. Although previous work used the description by data matrix, to simplify the comparison with other methods, I use the description of the blending operator in the time domain. Furthermore, since this is an inversion deblending technique that combines a least-squares solution with noise filters, this technique is perfect for a first experiment with seismic deblending, due to its simplicity.

[24] it does not have a gradient

3.4.1 Description

The iterative deblending, introduced by Mahdad *et al.* (2011), aims to solve the deblending problem by imposing sparsity without directly minimizing equation 3.8:

$$\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{T}\mathbf{x}\|_2^2 + \lambda \|\mathbf{S}\mathbf{x}\|_0 .$$

In fact, he proposed to minimize a slightly different objective function given by:

$$\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{T}\mathbf{x}\|_2^2 \text{ subject to } \mathbf{x} \in R(F), \tag{3.9}$$

where F is a coherence filter[25], and $R(F)$ is its range[26].

Chen *et al.* (2014b) later introduced a general framework via shaping regularization to iteratively deblend seismic data. However, regardless of the iterative approach, in both cases, they combine the least-squares solution with filtering operators to impose sparsity before estimating a solution.

Equation 3.9 replaces the ℓ_0-norm term by a coherence filter applied to a partial solution in each iteration. Since the coherence filter (see Section 3.4.2 for more details) removes the incoherent blending noise in the data while preserving part of the signal, it increases the sparsity of the solution after each iteration.

The first step to build the iterative solution proposed by Mahdad *et al.* (2011) is to apply the pseudo-deblending operator ($\mathbf{T}^H$) to decompress and decode the blended seismic data (as illustrated in Figure 3.1). Equation 3.10 estimates $\mathbf{x}_0$, the input for the iterative solution:

$$\mathbf{x}_0 = \mathbf{T}^H\mathbf{b}. \tag{3.10}$$

Applying a coherence filter F on $\mathbf{x}_0$, it is possible to eliminate the blending noise and recover part of $\mathbf{x}$. Since only part of the solution is recovered in each iteration, a portion of the signal is removed along with the noise. Equation 3.11 gives an estimate of $\mathbf{x}_{0_F}$, the filtered solution:

$$\mathbf{x}_{0_F} = F(\mathbf{x}_0). \tag{3.11}$$

[25] a coherence filter is a combination of filtering operators designed to remove the incoherent noise
[26] it is the space of all possible solutions generated with the application of the filter

From the partial filtered solution, it is possible to compute the blending noise that would be generated by that data (reblending $\mathbf{x}_{0_F}$, pseudo-deblending it, and subtracting it from $\mathbf{x}_{0_F}$ itself). Equation 3.12 shows how to estimate the noise:

$$\mathbf{x}_{0_N} = \mathbf{T}^H \mathbf{T} \mathbf{x}_{0_F} - \mathbf{x}_{0_F}. \tag{3.12}$$

Finally, the solution for the first iteration is the subtraction of the blending noise ($\mathbf{x}_{0_N}$) from $\mathbf{x}_0$, which gives the pseudo-deblended solution with a portion of the blending noise attenuated

$$\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{x}_{0_N}. \tag{3.13}$$

Repeating this procedure, decreasing the filter attenuation (to allow more signal to enter into the partial solution) is possible to recover fully deblending data. The complete iterative solution for equation 3.9 is given by:

$$\mathbf{x}_{i+1} = \mathbf{T}^H \mathbf{b} - [\mathbf{T}^H \mathbf{T} - I] F(\mathbf{x}_i). \tag{3.14}$$

In practice, this process is not perfect, and it is not uncommon that part of the blending noise still contaminates the final solution. Figure 3.2 illustrates the steps described by equations 3.10 to 3.13.

3.4.2 Filtering Applied to Deblending

A key factor that affects the quality of the final deblended data is the coherence filter. The coherence filter, mentioned before, is a composition of denoising filters with the objective of separating signal from noise.

Standard filtering techniques for deblending applications are:

- **F-K filter** — Based on filtering by dipping events, the F-K filter takes advantage of the characteristic of how coherent and random events are mapped in the frequency-wavenumber domain (F-K). In general random events are spread over all wavenumbers, while coherent events are confined in a specific region.

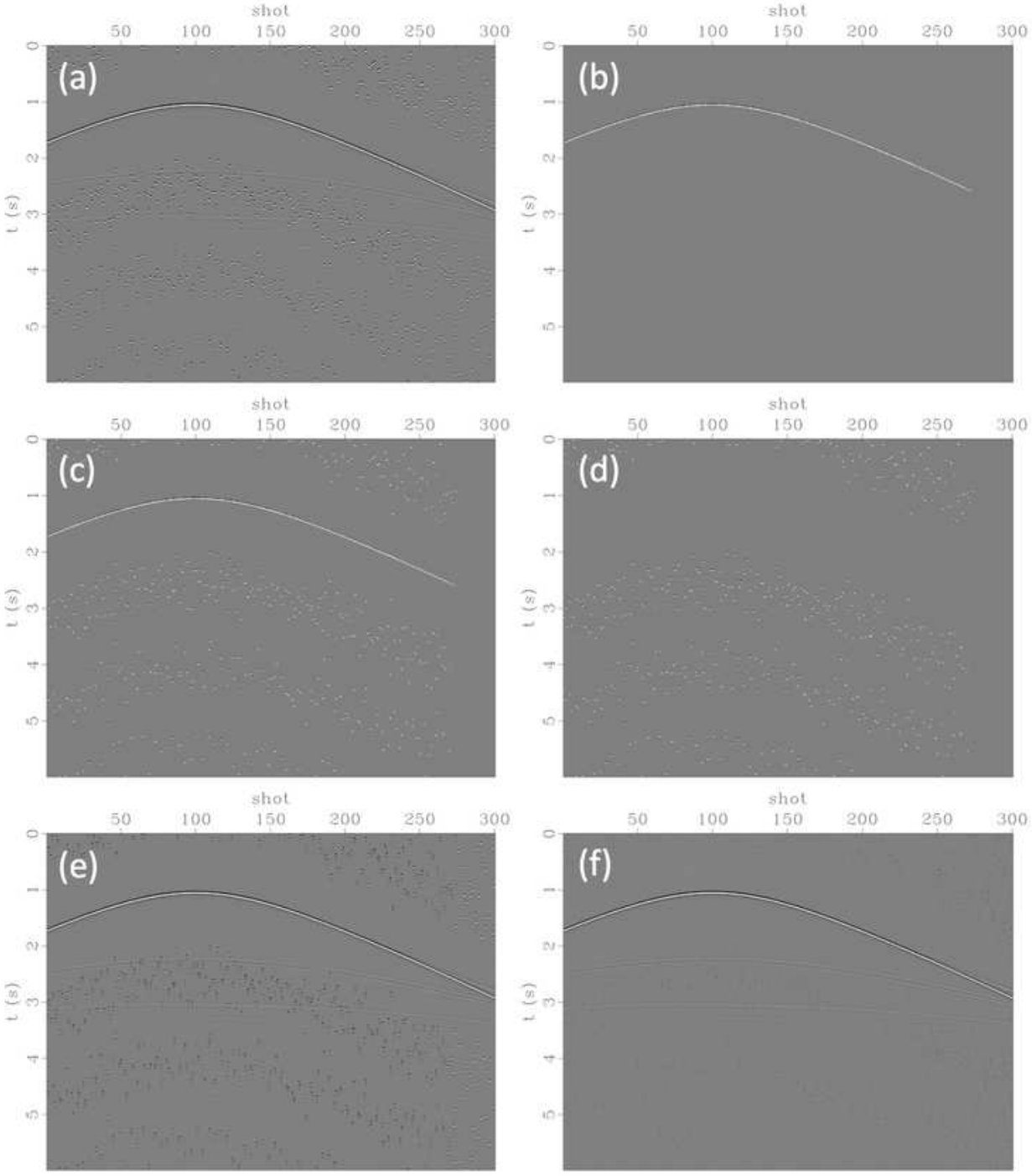

Figure 3.2: Iterative deblending in the common receiver domain. Pseudo-deblended data (a), $\mathbf{x}_0 = \mathbf{T}^H\mathbf{b}$. Filtered data (b), $F(\mathbf{x}_0)$. Reblended and pseudo-deblend data (c), $\mathbf{T}^H\mathbf{T}\mathbf{x}_{0_F}$. Estimated blending noise (d), plot c $-$ plot b. Deblended data after one iteration (e), and deblended data after 20 iterations (f).

The filtering step consists of measuring the apparent velocities (dips in the time-space domain) and setting areas of the F-K spectrum to be preserved or removed, following the equation

$$V_{ap} = \lambda f = \frac{2\pi f}{k_x}, \tag{3.15}$$

where V_{ap} is the apparent velocity , λ is the wavelenght, f is the frequency, and k_x is the wavenumber. Setting limits on the f-k spectrum filter velocities below V_{ap}.

The discrete Fourier transform requires a regular sampling interval, which is not the case for field data, that has non-uniform spatial sampling. Because of that, noise is introduced in this process. However, for marine seismic acquisitions, the spatial variations are minimal, and the noise is negligible. In cases where the spatial variations are relevant, a non-uniform discrete Fourier transform can be used;

- **Median filter** — Non-linear filter that for each sample of the data, evaluates the median within a window of neighbor samples, and replaces the original amplitude by it;

- **Time-Frequency Denoise filter (TFDN)** — Windowed median filter applied in the frequency domain. For each sample in the time domain, a 2D window is defined, then transformed to the frequency domain, where for each frequency the median is evaluated and compared with a threshold value to discriminate if an anomalous amplitude should or not be replaced by the median (Elboth *et al.*, 2008);

- **Sparse transforms filtering** — Sparse transforms, such as Wavelets and Curvelets, provides a sparse representation of seismic data (Antonini *et al.*, 1992; Candes & Donoho, 2000) that combined with a non-linear reconstruction[27] becomes a powerful denoising filter.

[27] keeping only the k largest coefficients in magnitude, where k is much less than the number of coefficients

For marine seismic data blended with a linear phase encoding scheme, the signal and blending noise amplitudes are similar. This makes it necessary to combine denoising filters with thresholding that removes the remaining noise by discriminating high and low amplitudes.

Thresholdings are defined in time, frequency or another sparse domain, and they can be *hard* or *soft*. A *hard-thresholding* sets all samples with amplitudes bellow to some value to zero. While a *soft-thresholding* uses a taper to linearly decrease the amplitudes above to some value, preventing discontinuities.

Equation 3.16 and 3.17 show the definition of hard and soft thresholds in the time domain:

$$Th_{Hard}(x) = \begin{cases} x\ ,\ \|x\| \geq \tau_h \\ 0\ ,\ \|x\| < \tau_h \end{cases} , \tag{3.16}$$

$$Th_{Soft}(x) = \begin{cases} x - \tau_h\ ,\ x \geq \tau_h \\ x + \tau_h\ ,\ x \leq -\tau_h \\ 0\qquad ,\ -\tau_h < x < \tau_h \end{cases} , \tag{3.17}$$

where τ_h is the amplitude thresholding.

For deblending applications, τ_h is set to decrease exponentially with the number of iterations. This is, $\tau_h = A \times a^i$, where A is the maximum amplitude, a is a number smaller than 1, and i is the number of the iteration.

Figure 3.3 shows the FK spectra of a receiver gather before and after applying the FK filter and the TFDN filter. Because of the randomness of the blending noise, it is spread across all wavenumbers (Figure 3.3b). Since the region outside of the F-K cone does not contain useful signal, a velocity filter in F-K domain was applied (Figure 3.3c). Additionally, a TFDN filter followed by a hard threshold recovers the predicted signal for one specific iteration (Figure 3.3d).

More details about the parametrization of these filters are available in Chapter 4.

3.4.3 Stop Condition

Any iterative solution needs a *stop* or *convergence criteria*. For the iterative deblending, a simple and effective stop condition consists of applying the blending operator to the output

of each iteration and comparing with the original blended data. This is done by extracting the RMS amplitude of their difference:

$$\text{RMS Error}_i = \sqrt{\sum_{n=1}^{N} \frac{(\mathbf{T}x_{i_n} - b_n)^2}{N^2}}, \tag{3.18}$$

where N is the number of samples of the blended data, and i is the current iteration.

Considering that in each iteration the *RMS Error* needs to decrease, by setting a limit value ($\text{RMS Error}_i < \alpha$), it is possible to iterate until the required level of accuracy is obtained.

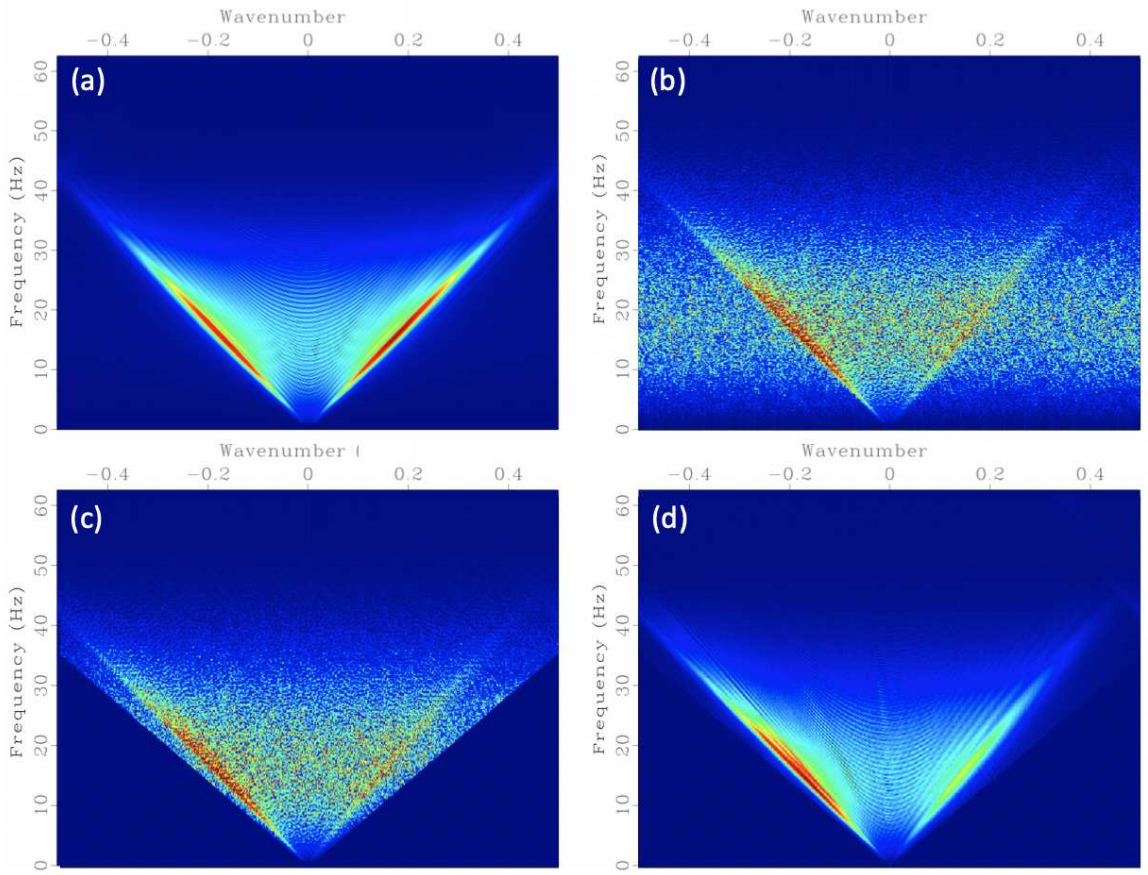

Figure 3.3: Coherence filters FK spectra. Unblended receiver gather (a), pseudo-deblended receiver gather (b), pseudo-deblended receiver gather after F-K filtering (c), and pseudo-deblended receiver gather after F-K filtering, TFDN filtering and a hard threshold (d).

3.5 Deblending by Compressive sensing

Compressive sensing (also known as compressed sensing, or only CS), introduced by Donoho (2006) and Candes *et al.* (2006), is a compelling technique to acquire random sub-Nyquist sampling signals and reconstruct[28] using prior knowledge that the data is sparse in some domain.

A direct benefit of compressive sensing, when applied to seismic acquisitions, is the possibility of acquiring data more sparsely, thus, allowing seismic surveys to be acquired at lower costs. Although the combination of random spatial sampling and blended surveys has excellent potential for cost reduction, in this project, a quasi-regular[29] spatial sampling with the random time-dithering scheme are considered. In this case, although the shooting and receivers positions are not random, the compressive sensing technique can be applied for seismic deblending.

3.5.1 Description

The deblending by compressive sensing is described in the literature by different authors (Kumar *et al.*, 2015b; Wason, 2017). Similar to iterative deblending, the method imposes sparsity without solving the objective function given by equation 3.7:

$$\min_{\mathbf{x}} \|\mathbf{Sx}\|_0 \text{ subject to } \mathbf{b} = \mathbf{Tx},$$

Going back to the definition of the blended and deblended data vectors ($\mathbf{b}$ and $\mathbf{x}$, respectively) in Chapter 2, we know that $\mathbf{b}$ has size n and $\mathbf{x}$ has size N, where $n \ll N$. Considering the solution to our problem (deblended data) being sparse in some domain, this means that only a few elements of $\mathbf{Sx}$ are necessary to reconstruct $\mathbf{x}$. Mathematically, the sparsity k of a vector is calculated by evaluating its ℓ_0-norm ($k = \|\mathbf{Sx}\|_0$). However, as mentioned earlier, the ℓ_0-norm is a combinatorial problem, which means that the number of possible solutions increase rapidly with the size of the data. Based on that, alternative procedures

[28] by solving underdetermined linear systems
[29] small spatial variations from the acquisition pre-plot are allowed

for obtaining sparse solutions must be used. Instead of solving equation 3.7, the compressive sensing approach proposes to replace the ℓ_0-norm by the ℓ_1-norm[30]:

$$\min_{\mathbf{x}} \|\mathbf{Sx}\|_1 \text{ subject to } \mathbf{b} = \mathbf{Tx}, \tag{3.19}$$

where $\|\mathbf{Sx}\|_1$ is the sum of the magnitude of the elements in $\mathbf{Sx}$. This description is called the *Basis pursuit problem*.

The Basis pursuit (or only BP) is a convex optimization problem employed when the relation $\mathbf{b} = \mathbf{Tx}$ must be satisfied exactly. Because of this, the method can not handle noisy data. For seismic applications a extension of BP method is often used, the *Basis-pursuit denoising* or BPDN. Equation 3.20 describes the same problem of equation 3.19, but now with sensitivity to noise (BPDN formulation):

$$\min_{\mathbf{x}} \|\mathbf{Sx}\|_1 \text{ subject to } \|\mathbf{b} - \mathbf{Tx}\|_2 \leq \sigma, \tag{3.20}$$

where $\mathbf{S}$ is a sparse transform, and σ is the misfit level that sets the noise level allowed into the solution.

In practical terms, equation 3.20 can be rewritten in terms of the data vector in the sparse transform domain ($\mathbf{f} = \mathbf{Sx}$), which simplifies the deblending algorithm implementation:

$$\min_{\mathbf{f}} \|\mathbf{f}\|_1 \text{ subject to } \left\|\mathbf{b} - \mathbf{TS}^H\mathbf{f}\right\|_2 \leq \sigma, \tag{3.21}$$

where $\mathbf{S}^H$ is the inverse transform.

After solving equation 3.21, the final deblended data is recovered by applying the inverse transform $\mathbf{x} = \mathbf{S}^H\mathbf{f}$. The CS deblending algorithm is available in the appendix.

Since the deblending problem was written as a Basis-pursuit denoising optimization problem, solutions available in the literature can be used.

Van Ewout Berg & Friedlander (2008) introduced a root-finding algorithm on the Pareto curve that solves BPDN problems available through the solver $SPG\ell_1$[31]. Yang & Zhang (2011) use a different approach, available through the solver YALL1[32], to reformulate an ℓ_1-

[30] due to the convex geometry of the ℓ_1-norm, the ℓ_1-norm minimization generates sparse solutions
[31] `https://friedlander.io/spgl1`
[32] `http://yall1.blogs.rice.edu/`

problem into separable objective functions to apply an alternating direction method. Becker *et al.* (2011) use a smoothing technique to improve the convergence of standard gradient-descent algorithms. Their algorithm is available through the solver NESTA[33].

A comparison between some of the available solvers was carried out by Le Montagner *et al.* (2011), and according to their tests, the solvers generate similar results, although the performance is different. In practice, the best solver is the one that fulfills the problem requirements. In this project, I have tested the solvers available for the Python[34] language, and the $SPG\ell_1$ was the most flexible in working in real and complex domains, and with linear operators, which is a requirement for the CS deblending implementation.

3.5.2 Lasso convex optimization problem - Spectral projected-gradient

As introduced in Section 3.5.1, the BPDN formulation is the basis of the CS deblending, and a possible way to solve a BPDN optimization problem is using the $SPG\ell_1$ algorithm. However, as will be made clear in Section 3.5.3, the solution of a BPDN problem relies on the ability to solve another problem: a *Lasso Optimization problem*. A Lasso problem is formulated as:

$$\min_{\mathbf{x}} \|\mathbf{b} - \mathbf{T}\mathbf{x}\|_2^2 \text{ subject to } \|\mathbf{S}\mathbf{x}\|_1 \leq \tau \tag{3.22}$$

where τ is the maximum allowable magnitude[35] for $\|\mathbf{S}\mathbf{x}\|_1$.

Before describing the $SPG\ell_1$ algorithm to solve BPDN problems, a method for solving Lasso problems is described. This method is the basis for the $SPG\ell_1$ algorithm.

Lasso and BPDN optimization problems are both equivalent:

$$\text{LASSO: } \min_{\mathbf{x}} \left\|\mathbf{b} - \mathbf{T}\mathbf{S}^H\mathbf{f}\right\|_2 \text{ subject to } \|\mathbf{f}\|_1 \leq \tau,$$

$$\text{BPDN: } \min_{\mathbf{f}} \|\mathbf{f}\|_1 \text{ subject to } \left\|\mathbf{b} - \mathbf{T}\mathbf{S}^H\mathbf{f}\right\|_2 \leq \sigma,$$

[33] `https://statweb.stanford.edu/~candes/software/nesta/`

[34] `https://www.python.org/`

[35] τ is related to how sparse the solution is, the selection of its magnitude is not as straighfoward as defining the number of desired non-zero coefficients in the final solution, since is data dependent.

which means they return similar solutions as far the σ and τ are selected appropriated. However, the relation between these parameters is not straightforward.

As will be detailed in Section 3.5.3, the solution for a BPDN problem with a particular σ can be obtained by solving sequences of Lasso problems. For this goal, Birgin *et al.* (2000) introduced an algorithm to solve Lasso Problems, called spectral projected-gradient (SPG), which is very effective for large problems.

The SPG method relies on a projection operator

$$P_\tau[c] = \left\{ arg\min_f \|\mathbf{c} - \mathbf{f}\|_2 \text{ subject to } \|\mathbf{f}\|_1 \leq \tau \right\}, \tag{3.23}$$

that iterates the set $\left\{ \mathbf{f} \mid \|\mathbf{f}\|_1 \leq \tau \right\}$ of possible solutions.

Initially, the SPG projects $\mathbf{f_0}$, which is the initial guess (if available) or *zero*: $\mathbf{f_l} = P_\tau[\mathbf{f_0}]$. Then, it evaluates the residual $\mathbf{r} = \mathbf{b} - \mathbf{TS}^H\mathbf{f_l}$ and its gradient $\mathbf{g_l} = -\mathbf{TS}^H\mathbf{r}$. Based on $\mathbf{f_l}$ and $\mathbf{g_l}$, the algorithm iteratively searches the projected gradient path $\bar{\mathbf{f}} = P_\tau[\mathbf{f_l} - \alpha\mathbf{g_l}]$ until the relative residual $\bar{\mathbf{r}} = \mathbf{b} - \mathbf{TS}^H\bar{\mathbf{f}}$ decreases less than $\gamma(\bar{\mathbf{f}} - \mathbf{f}_l)$, where α is the step length and γ is the desired descent parameter. When this condition is reached $\mathbf{f}_{l+1} = \bar{\mathbf{f}}$, $\mathbf{r} = \bar{\mathbf{r}}$, and $\mathbf{g_{l+1}} = -\mathbf{TS}^H\mathbf{r}$, and the process is repeated until convergence is achieved. The complete algorithm can be found in Birgin *et al.* (2000).

3.5.3 BPDN Convex Optimization Problem - SPGl1 Solver Overview

Van Ewout Berg & Friedlander (2008) proposed the $SPG\ell_1$ algorithm to solve BPDN problems, which reduces it to a problem of finding the root (based on Newton's method) of a single parameter nonlinear equation. For this, the algorithm evaluates a sequence of Lasso problems for different τ's and estimates an L1-curve (Pareto Curve). Then, evaluating the derivative of the L1-curve and extrapolating for the desired misfit σ, the algorithm iteratively obtains the optimum τ that recovers the signal.

Assume that a Lasso problem for a specific τ can be solved using the spectral projected-gradient (SPG) algorithm. Then, a solution $\mathbf{f}_\tau$ is recovered. From this, a single parameter

equation, parameterized by τ, can be written as:

$$\phi(\tau) = \|\mathbf{r}_\tau\|_2 = \sigma, \tag{3.24}$$

where $\mathbf{r}_\tau = \mathbf{b} - \mathbf{T}\mathbf{S}^H\mathbf{f}_\tau$.

The function ϕ is called the *L1-curve* or *Pareto curve*, and relates the two-norm of the residual ($\left\|\mathbf{b} - \mathbf{T}\mathbf{S}^H\mathbf{f}\right\|_2$) and the one norm of the solution ($\|\mathbf{f}_{\tau_k}\|_1$).

Since it can be proved that the L1-curve is a convex and continuously differentiable function, it is possible to find the root of equation 3.24 using a Newton-based algorithm, where ϕ and ϕ' at some τ are required. In practice, the complete L1-curve is unknown, but still, the iterative method is able to find its root.:

$$\tau_{i+1} = \tau_i - \frac{\phi(\tau_i)}{\phi'(\tau_i)}. \tag{3.25}$$

Figure 3.4, adapted from van den Berg & Friedlander (2019), shows an L1-curve and illustrates the root-finding algorithm. After setting $\tau = \tau_0$, the SPG algorithm is applied to solve the Lasso problem, and $\phi(\tau_0)$ and its gradient - $\phi'(\tau_0)$ - are calculated (Figure 3.4a). Then, extrapolating the gradient until the desired σ (red dashed-line) is possible to find a new $\tau = \tau_1$. Then, evaluating $\phi(\tau_1)$ (Figure 3.4b) and $\phi'(\tau_1)$ (Figure 3.4c) is possible to repeat the process until $\phi(\tau_n) = \sigma$, and the BPDN solution is found.

From the user perspective, the $SPG\ell_1$ solver requires almost no intervention. For the BPDN problem the following information is required:

- The blending operator $\mathbf{A} = \mathbf{T}\mathbf{S}^H$;
- The blended data $\mathbf{b}$;
- The misfit limit σ .

Because of its flexibility in accepting operators, any sparse transform available for Python language can be used, including Fourier, Wavelet, and Curvelet transform.

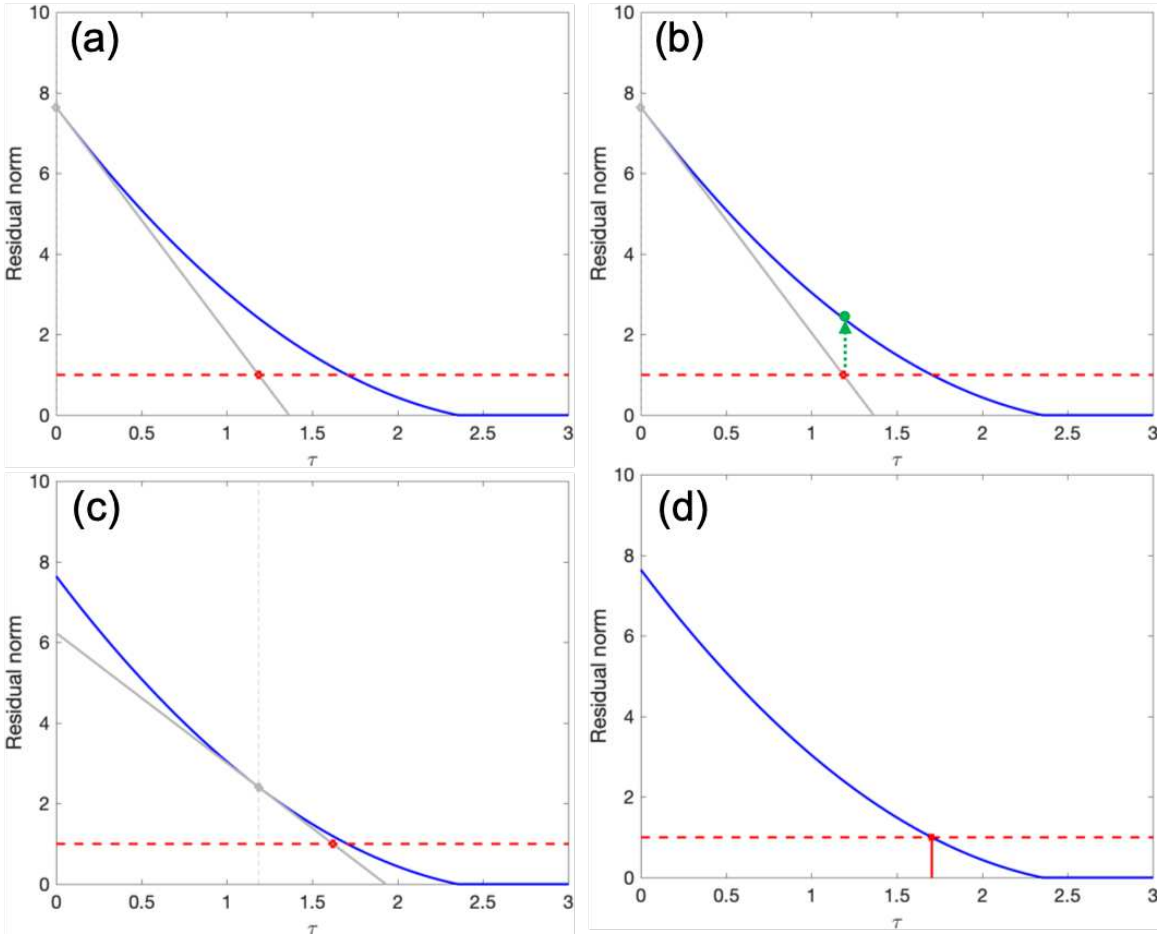

Figure 3.4: Pareto curve and Newton's method solution. a) The first iteration, after applying the SPG algorithm, Newton's method, and extrapolating to the σ-line (red dashed line) . b) Beginning of the 2nd iteration with the evaluation of ϕ using the SPG algorithm. c) Evaluation of ϕ' and extrapolation to the σ-line (still 2nd iteration) d) Final iteration, where $\phi(\tau_n) = \sigma$ and the BPDN solution is found. Adapted from van den Berg & Friedlander (2019).

3.5.4 Sparse Recovery - Conditions and Analysis

Although the methods presented above are useful for recovering sparse solutions, a question that remains is: under which conditions is the sparse solution recoverable and unique?

As described before, the deblending problem is underdetermined, which means $\mathbf{T}$ is not invertible, and not even square. Additionally, from the problem description, we also know that the columns of $\mathbf{T}$ are not mutually orthogonal. If they were, the deblended solution would be unique.

According to Candes *et al.* (2008), given the above conditions, it is possible to reconstruct a signal with good accuracy if:

1. the signal $\mathbf{f}$ is sufficiently sparse;
2. the matrix $\mathbf{A} = \mathbf{TS^H}$ obeys a condition known as the restricted isometry property (RIP), which means, submatrices of $\mathbf{A}$ behave nearly as an orthogonal basis, when operating on $\mathbf{f}$.

To demonstrate the above statements, Candes *et al.* (2008) use the concept of restricted isometry constants, where it is possible to define a constant δ_k, for all integers k, as the smallest number such that

$$(1 - \delta_k) \left\|\mathbf{f_k}\right\|_2^2 \leq \left\|\mathbf{T}_k \mathbf{S}^H \mathbf{f_k}\right\|_2^2 \leq (1 + \delta_k) \left\|\mathbf{f_k}\right\|_2^2 \tag{3.26}$$

is valid for all sparse vectors $\mathbf{f}_k$, where $\mathbf{T}_k$ is a submatrix of $\mathbf{k}$ formed by any set of k columns of $\mathbf{T}$.

From equation 3.26 is possible to show that when $\delta_{2k} < 1$ the solution is unique, and when $\delta_{2k} < \sqrt{2} - 1$ the solution for the $\ell_1 - norm$ is equivalent to the $\ell_0 - norm$.

In practice, for seismic deblending applications, numerical analysis on synthetic data can be performed to demonstrate the level of sparsity necessary to recover good deblended data.

To quantitatively analyze the sparsity of a representation, a simple and useful approach is generating a plot of the energy of a reconstructed signal considering only a fraction of the

largest coefficients in the transform domain. Figure 3.5 shows the reconstruction analysis for 3 different transforms: Fourier, Wavelet and Curvelet transfom. A sail line with 300 shots, 300 receivers, and 1,500 time samples were extracted from a field data.

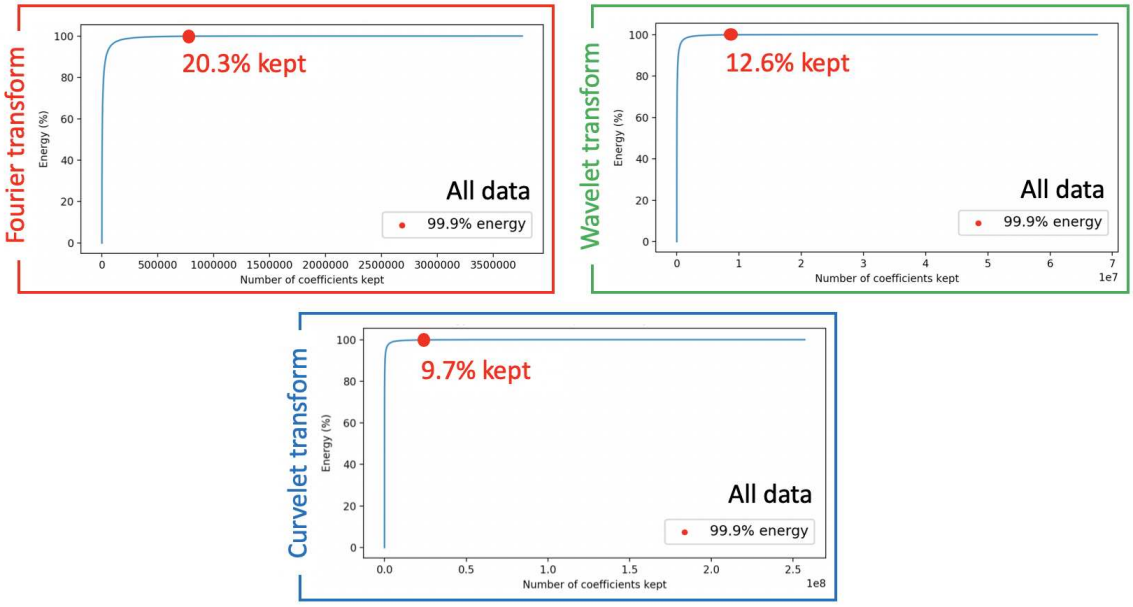

Figure 3.5: Field data sparsity analysis with 3D Fourier, 3D Wavelet, and 3D Curvelet transform on a sail line.

The objective of this analysis is to find the minimum number of coefficients necessary to reconstruct the signal with 99.9% of accuracy. Fewer coefficients mean a more sparse representation. The reconstruction energy is calculated using:

$$Energy(\%) = \left(1 - \frac{\left\|\mathbf{f} - \tilde{\mathbf{f}}_k\right\|_2^2}{\|\mathbf{f}\|_2^2}\right) \times 100\% \tag{3.27}$$

where $\mathbf{f}$ is the original data, and $\tilde{\mathbf{f}}_k$ is the reconstructed signal with only the k-largest Fourier coefficients.

The reconstruction of the sail line, with 99.9% of the energy, requires 20.3% of the coefficients in the Fourier domain, 12.6% of the coefficients in the Wavelet domain, and 9.7% of the coefficients in the Curvelet domain. In this case, the Curvelet is more sparse than the others transforms.

Because of the size of the seismic data, applying 3D transforms often requires a large amount of memory. In those cases, 2D transforms into different domains are used. However, it is crucial to understand how it affects the data sparsity.

Repeating the above procedure, but now considering the reconstruction of a common receiver gather, it is possible to analyze how sparse those gathers are. Figure 3.6 shows the reconstruction analysis for 3 different transforms: Fourier, Wavelet and Curvelet transfom. A common receiver gather with 300 traces, and 1,500 time samples was extracted from the same dataset used before.

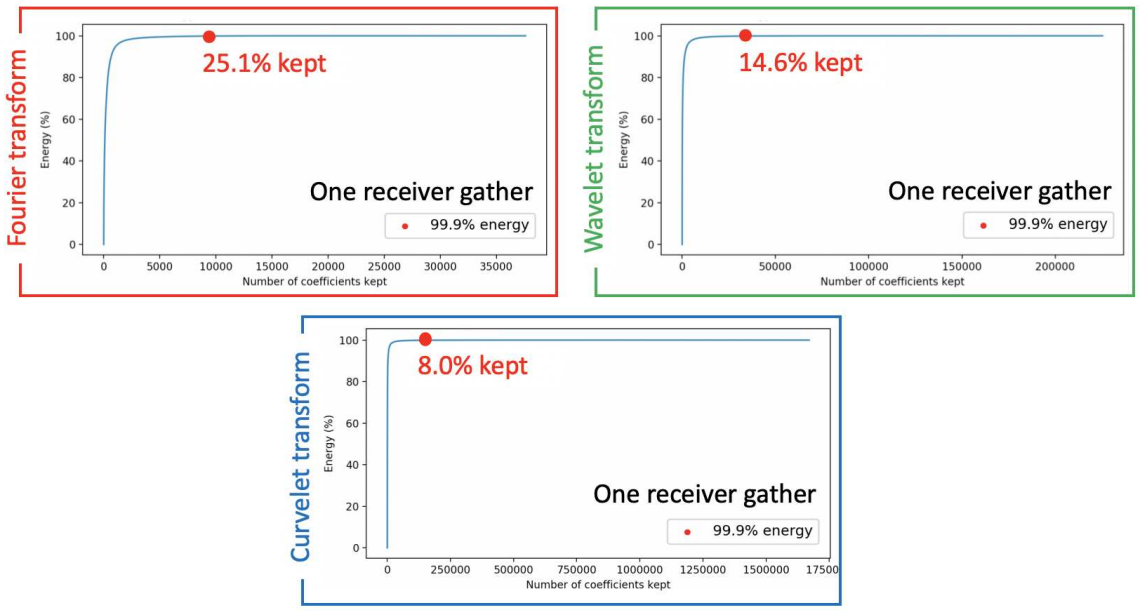

Figure 3.6: Field data sparsity analysis with 2D Fourier, 2D Wavelet, and 2D Curvelet transform on a common receiver gather.

The reconstruction of a common receiver gather, with 99.9% of the energy, requires 25.1% of the coefficients in the Fourier domain, 14.6% of the coefficients in the Wavelet domain, and 8.0% of the coefficients in the Curvelet domain. As before, the Curvelet is more sparse than the others transforms.

Based on the previous results, it is possible to conclude that, for example, deblending each common receiver gather independently in the Curvelet domain is more effective than

deblending the entire data simultaneously in the Fourier domain. To confirm the last statement, Figure 3.7 compares the results of the deblending by compressive sensing applied to each common receiver gather independently in the Curvelet domain(c), and the entire sail line in the Fourier domain (d).

3.6 Deblending Methods Remarks

In this chapter, I introduced the mathematical description of inversion based seismic deblending techniques. Since the goal with seismic deblending is to recover unblended records from blended data, an inverse problem was formulated, and possible solutions were discussed.

Initially, I tested the least-squares solution, called *pseudo-deblending*, which was formulated by evaluating the best fit for the observed data (equation 3.1). Through the results showed, it was clear that the least-square solution does not deblend the data properly, since blending noise is introduced into the solution.

After the initial test, I imposed extra constraints into the problem's objective function, in order to take advantage of the prior knowledge that the seismic data is sparse.

The first method with the sparsity constraint, called *Iterative deblending*, follows a workflow in the data domain based on equation 3.14. Sparsity is introduced by the use of a coherence filter to attenuate blending noise and recover deblending data. The combination of well-established filtering techniques (such as F-K filter, TFDN filter, median filter, and thresholding) is frequently used as a coherence filter.

The second method, called *deblending by compressive sensing*, uses the *Basis Pursuit Denoise* (BPDN) formulation (equation 3.20) to rewrite the problem's objective function. The method seeks for solutions using well-established algorithms such as the spectral projected-gradient for the ℓ_1-nom.

Since both methods rely on the sparsity of the solution, a sparsity analysis was performed in different transform domains, such as, Fourier, Wavelet, and Curvelet transforms. For seismic deblending applications., the results demonstrated that the Curvelet is more sparse than the Fourier and Wavelet transform.

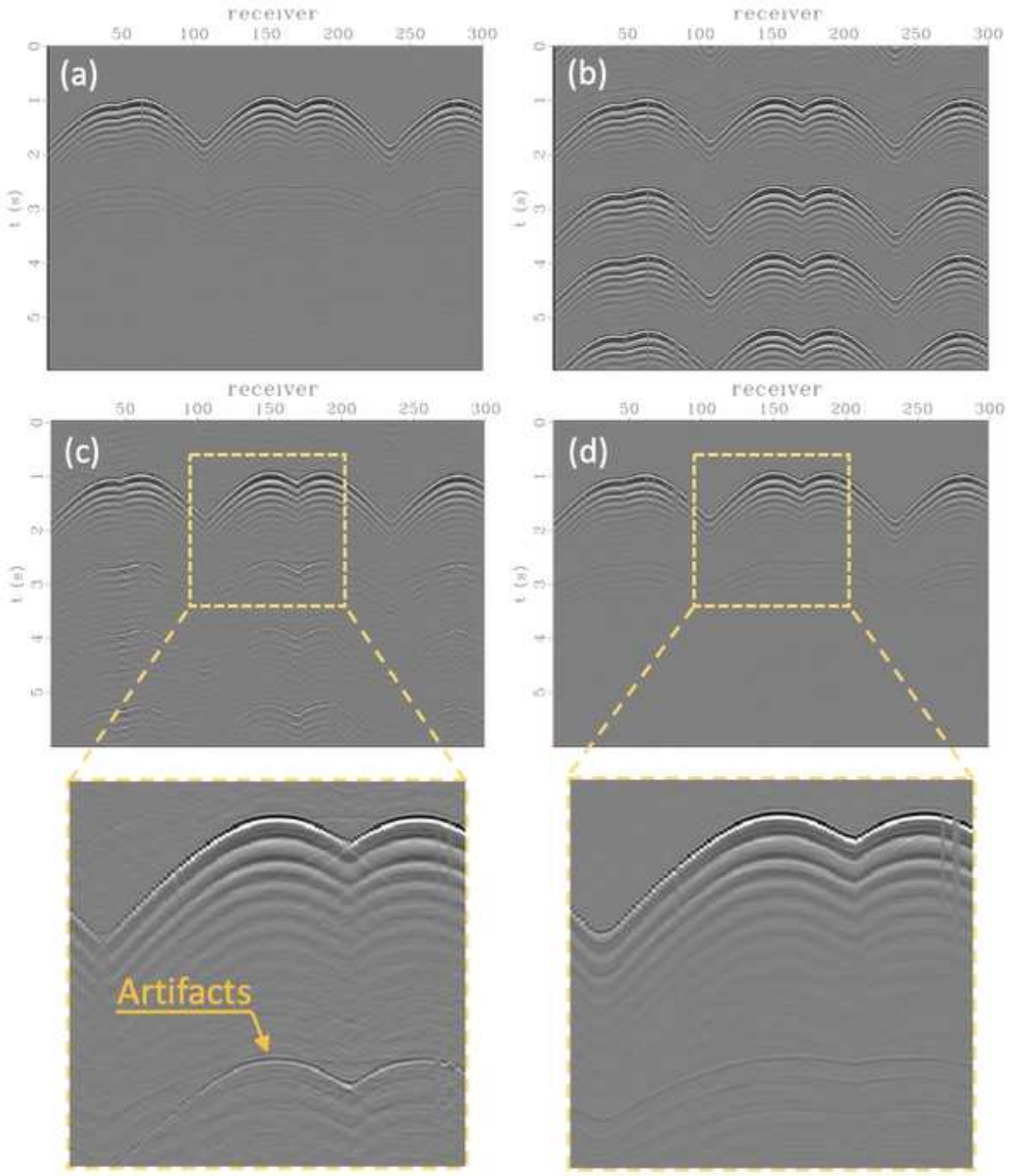

Figure 3.7: Comparison, in the shot domain, between deblending individual common receiver gathers in the Curvelet domain and deblending the entire sail line simultaneously in the Fourier domain. Original unblended data (a), blended data (b), deblended data (c) for individual common receiver gathers in the Curvelet domain, and deblended data (d) for the entire all sail line simultaneously in the Fourier domain. All plots have amplitude clipping to highlight the noise.

CHAPTER 4

DEBLENDING APPLICATION - SYNTHETIC DATA

In this chapter, I present the first deblending results. The primary analysis is carried out on synthetic pre-stack acoustic data simulating a time-lapse project, whose velocity and density models were generated based on properties extracted from a production field offshore Brazil.

For the analysis, I implemented the required algorithms to simulate a blended acquisition numerically and, using this data as input, performed the iterative deblending and the deblending by compressive sensing.

Since the objective of this thesis is to analyze not only the deblending methods but also their impact on 4D projects, I do quantitative analysis on the post-migration image domain. For this, RMS and NRMS amplitudes are extracted from some key reflectors to quantify the effect of residual blending noise on the 4D amplitude.

4.1 Seismic Modeling

In order to compare the deblending methods and their impact on 4D projects, I modeled two synthetic datasets simulating a time-lapse survey. As will be detailed in Chapter 5, Petrobras provided seismic and well log data acquired in the Jubarte field, located in the northern portion of the Campos basin (Figure 5.1a).

To make the tests with synthetic data more realistic, velocity and density information were extracted from the available data. Those properties were then interpolated onto a 2D grid, incorporating features found in the field, such as deepwater ocean bottom, post-salt reservoir, salt domes, and a pre-salt layer. Figure 4.1 shows the velocity and density models generated based on information extracted from the Jubarte field. This first model is the *baseline model*.

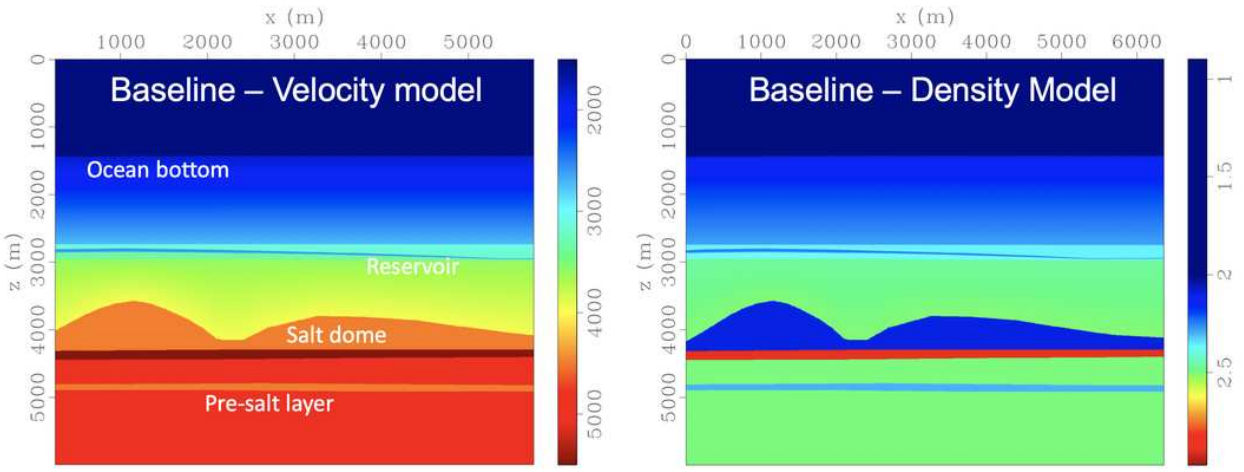

Figure 4.1: Baseline velocity and density models.

The seismic modeling used a 2D acoustic finite difference algorithm[36] available in the open-source package Madagascar (Fomel *et al.*, 2013). Figure 4.2 shows an illustration of the simulated geometry and Table 4.1 shows the main modeling parameters. The non-interference assumption between shots was considered, which means conventional unblended data were generated.

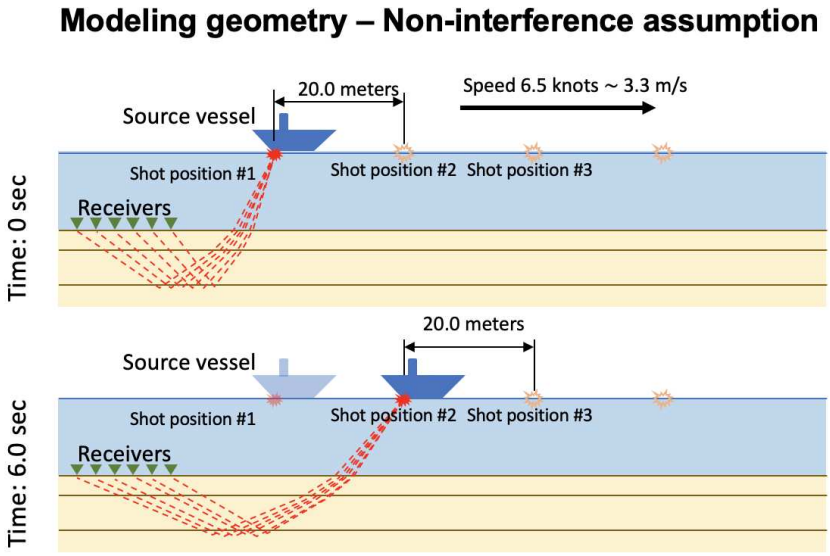

Figure 4.2: Modeling geometry.

[36]The 2D acoustic finite difference module is called *sfawefd2d*.

Table 4.1: Acoustic seismic modeling parameters.

	Parameters
Modeling grid (dx,dz,dt)	5 m, 5 m, 0.5 msec
Wavelet	Ricker
Dominant frequency	25 Hz
Source depth	10 m
Number of shots	300
Shooting interval	20 m
Receivers depth	1,430 m
Number of receivers	300
Receiver interval	20 m
Record length	6 sec
Final sample rate	4 msec

Using the methodology described above, I modeled 300 shot gathers. Figure 4.3a illustrates one of those shot gathers. The data were then *migrated* using a Reverse Time Migration (RTM) scheme (Levin, 1984). For the RTM, I used the same 2D finite difference modeling algorithm to back-propagate the receiver side wavefield. The back-propagated wavefield was then cross-correlated with the source wavefield and stacked to generate the final image. The exact velocity model[37] was used in the migration, this means that uncertainties related to the migration algorithm, and the velocity model were not introduced/considered. Figure 4.3b shows the migrated section for the baseline data.

Following the same methodology used for modeling the baseline data, I generated a velocity and density models for the 4D monitor, called *monitor model*. As before, the model followed the Jubarte Field properties, but now, production information was extracted and used to simulate production effects inside the post-salt reservoir.

Damasceno (2020) calibrated, for the Jubarte field, a rock physics model that relates the elastic properties of the rock to changes in fluid saturation and pore pressure. Based on this model, it was possible to determine how much an increase in water saturation from 20% to 80% would affect the acoustic properties of the reservoir (changes in pore pressure were not

[37]the velocity model has not been smoothed, as is usually done when the exact velocity model is not available.

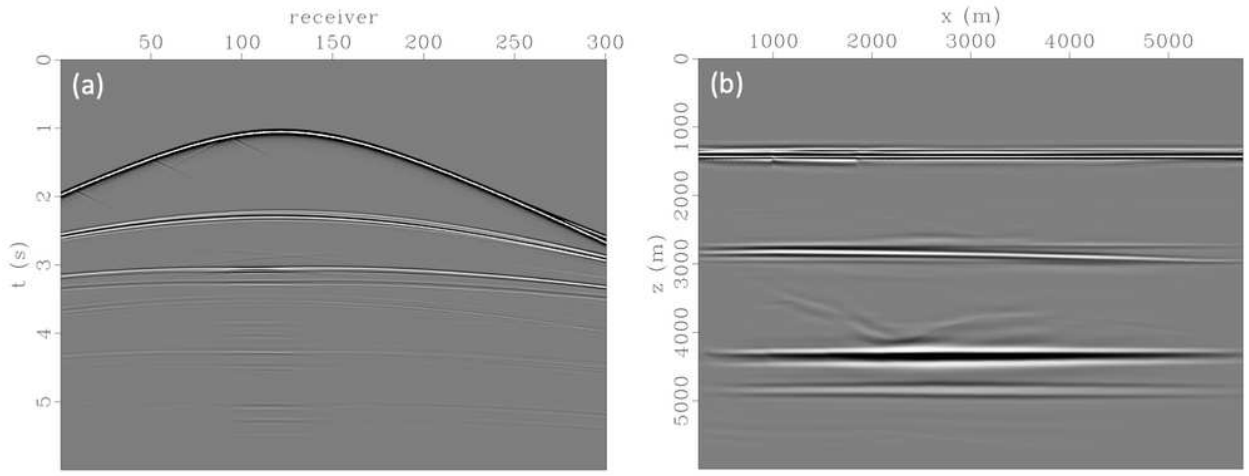

Figure 4.3: Baseline synthetic shot gather (a), and migrated section using a RTM algorithm (b). All plots have amplitude clipping.

considered).

Velocities inside the reservoir interval were increased by 4.51%, and the densities were increased by 1.13%. Figure 4.4 shows the *monitor* velocity and density models, and Figure 4.5 shows the introduced variations (on a different scale, to highlight the changes). Due to the small thickness of the reservoir (between 80 and 160 feet), significant kinematic variations are not expected in the modeled seismic data. On the other hand, the changes introduced in the models are sufficient to generate noticeable amplitude variations (dynamic effect).

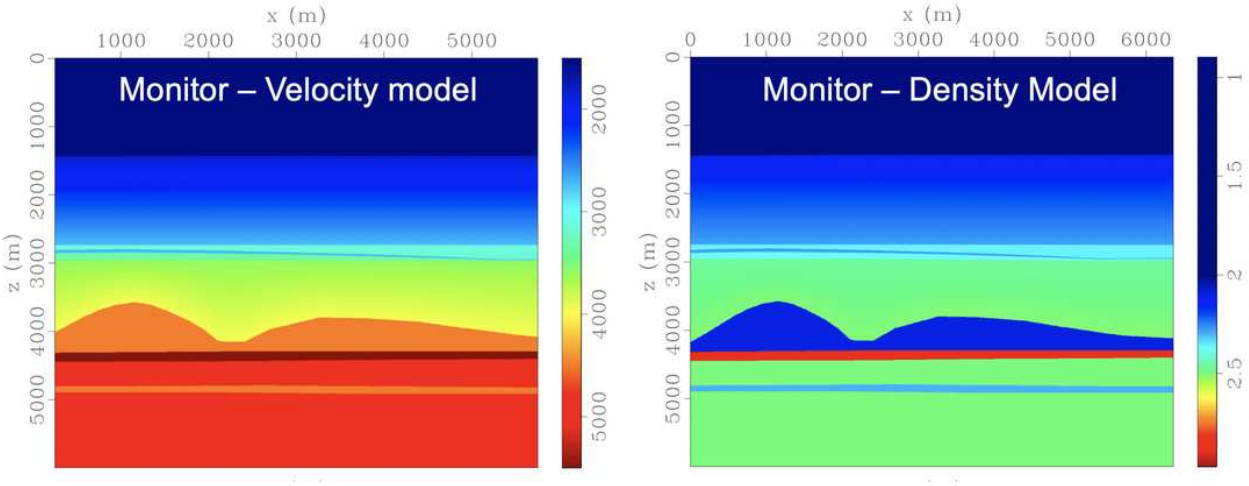

Figure 4.4: Monitor velocity and density models.

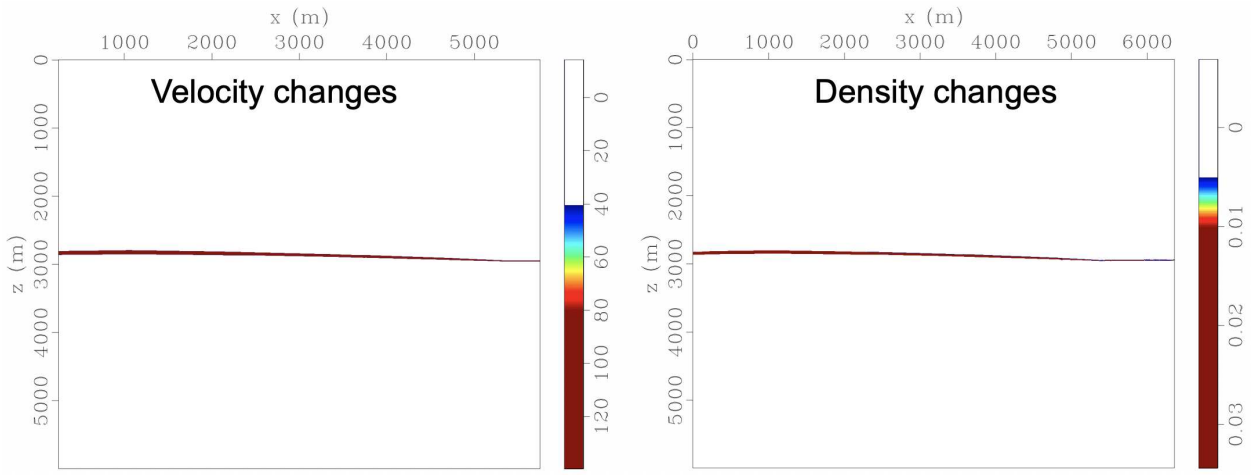

Figure 4.5: Variations introduced on the velocity and density models simulating productions effects.

Figure 4.6a shows a synthetic shot gather generated using the monitor model, with the same source position as used for the baseline (Figure 4.3a). As before, after modeling 300 shots, the data were *migrated* using the exact velocity model. Figure 4.6b shows the migrated section for the monitor synthetic data.

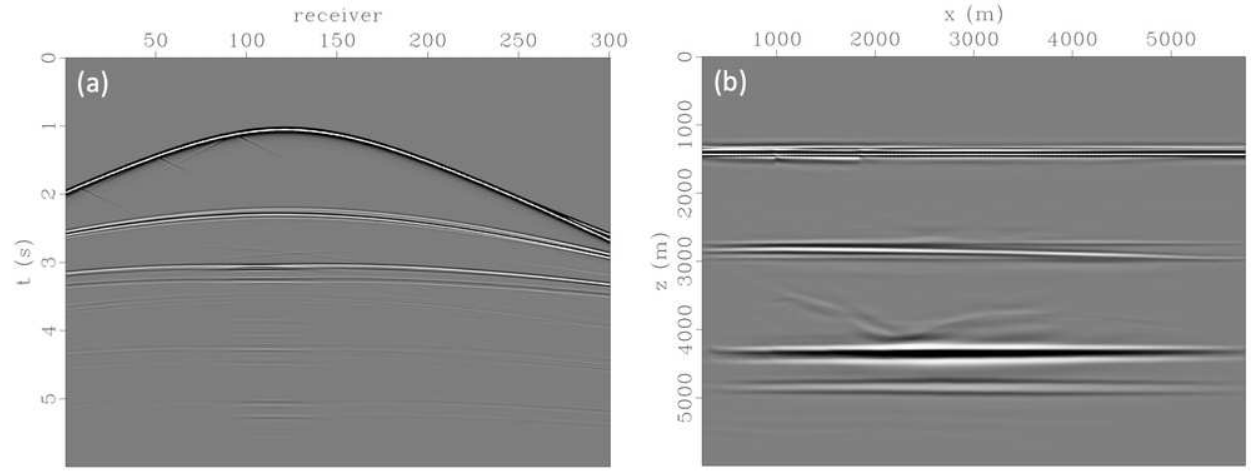

Figure 4.6: Monitor synthetic data. Modeled shot gather (a), and migrated section using an RTM algorithm (b).

Finally, it is possible to evaluate the 4D difference between the baseline and monitor data, which shows how the seismic data are affected by the simulated hydrocarbon production in

the reservoir. Figure 4.7 shows the 4D difference for our synthetic data. As expected, the amplitude difference in the reservoir is considerable, and outside the reservoir is very small. Changes in reservoir properties affect not only the reflection coefficients, but also the transmission coefficients. Time-shifts also occur, but, according to modeling carried out in the area, they are not relevant. These factors can influence differences in amplitudes in migrated sections, generating small 4D differences in reflectors below the reservoir interval.

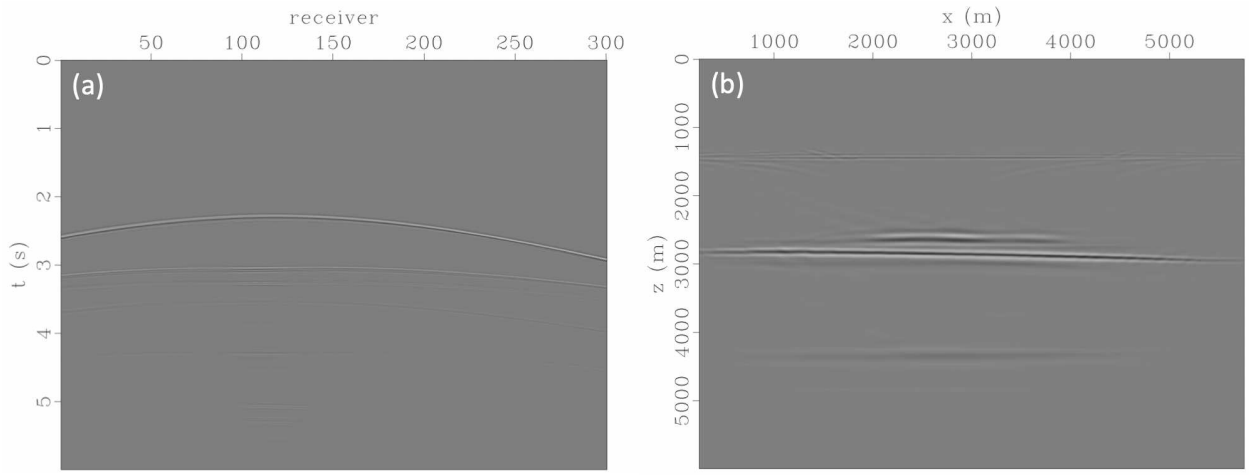

Figure 4.7: 4D difference (Monitor - Baseline) for the conventional unblended data. a) 4D difference in the shot domain, and b) 4D difference in the image domain. All plots have amplitude clipping.

To quantify the 4D amplitude differences, the NRMS amplitude is calculated. The NRMS is a measure of dissimilarity between datasets, and it is calculated inside a 2D window that encompasses the reflector to be analyzed. The NRMS is given by:

$$NRMS(\%) = \frac{2 \times RMS_{\text{4D difference}}}{RMS_{\text{Baseline}} + RMS_{\text{Monitor}}} \times 100\%, \tag{4.1}$$

where the RMS for a single trace is

$$RMS = \frac{\sqrt{\sum_{i=1}^{n} x_i^2}}{n}, \tag{4.2}$$

and n is the number of samples inside the window on the vertical direction.

A small NRMS value means less difference between the baseline and monitor datasets. Given that kinematic effects are insignificant. we expect a smaller NRMS in the pre-salt layer, and larger NRMS in the reservoir interval. Figure 4.8 shows the NRMS amplitudes for our synthetic datasets. The windows used for the NRMS calculation are show in the migrated section.

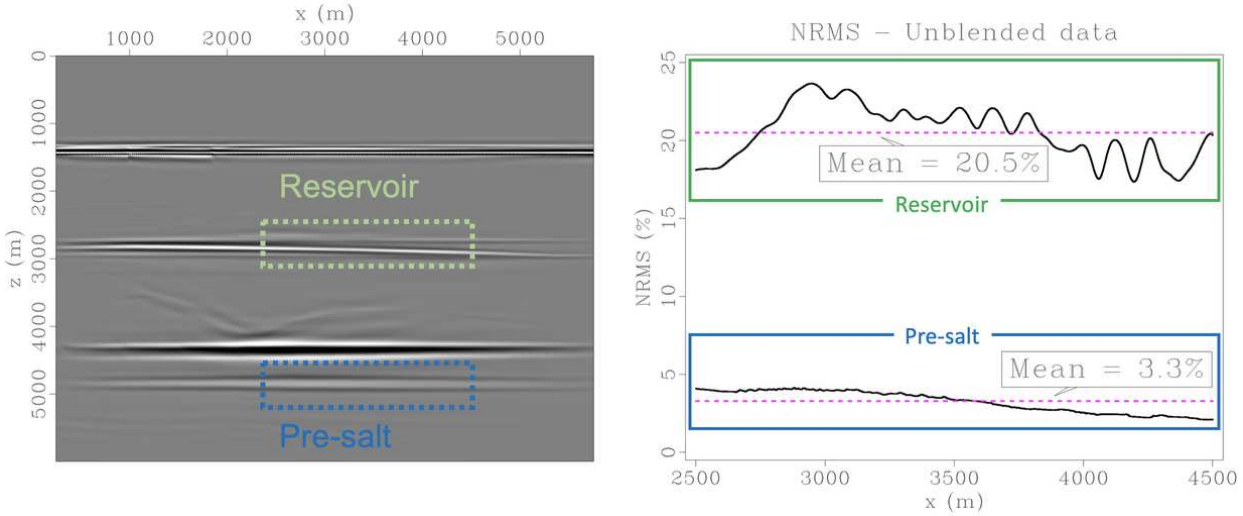

Figure 4.8: NRMS amplitude extracted from the conventional unblended data. The black line is the NRMS amplitude, and the pink dashed line is the mean value.

As we can see in Figure 4.8, inside the reservoir, we have a high NRMS. This is an expression of changes in reservoir properties in terms of NRMS. In the pre-salt layer we have a low, but not null, NRMS value. In this interval, as described, no 4D effects were introduced. Additionally, noise was not considered in this modeling, and the velocities and densities used in the modeling and migration are accurate. Therefore, the observed NRMS is due to transmission effects and small, uncorrected time-shifts. It affects all modeled data, so the approach that I use is to consider this NRMS as a background that affects all reflections bellow the reservoir.

4.2 Blending Simulation

Considering that the modeled seismic data followed the non-interference assumption, to use the data as a basis for this project, it was necessary to numerically blend the shots following the procedure described in Chapter 2. The advantage of blending them numerically, instead of directly modeling blended records, comes from the need to compare the deblending results with records generated without overlap.

As the main objective of this work is to simulate an operational gain, in terms of acquisition time and also in cost reduction, the simulation was carried out by overlapping unblended consecutive shots using equation 2.3. This procedure simulates an increase in the speed of the source vessel. The blending operator T follows the *random time-dithering scheme* (see Section 2.2). Simulating a source density ratio (SDR)[38] of 4, I introduced small random variations on the order of 0.2 seconds at firing intervals of 1.5 seconds, for the original 6.0 seconds records. These random variations are compatible with experimental results obtained by Aaron *et al.* (2016).

Figure 4.9 shows the blended geometry simulated and part of a blended continuous record generated from the unblended baseline data.

4.3 Deblending Results

Based on the *baseline* dataset, I first present results for the iterative deblending and then deblending by compressive sensing. The 4D analysis considering baseline and monitor data will be detailed in Section 4.4.

4.3.1 Iterative Deblending

As described in Section 3.4.2, the iterative deblending strongly relies on a coherence filter (equation 3.9). For the tests with the synthetic data, a combination of different filters were used to attenuate the blending noise in each iteration. The best combination included:

- F-K filter, filtering velocities below 1,500 m/s;

[38] which is equivalent to increase by 4 times the speed of the vessel

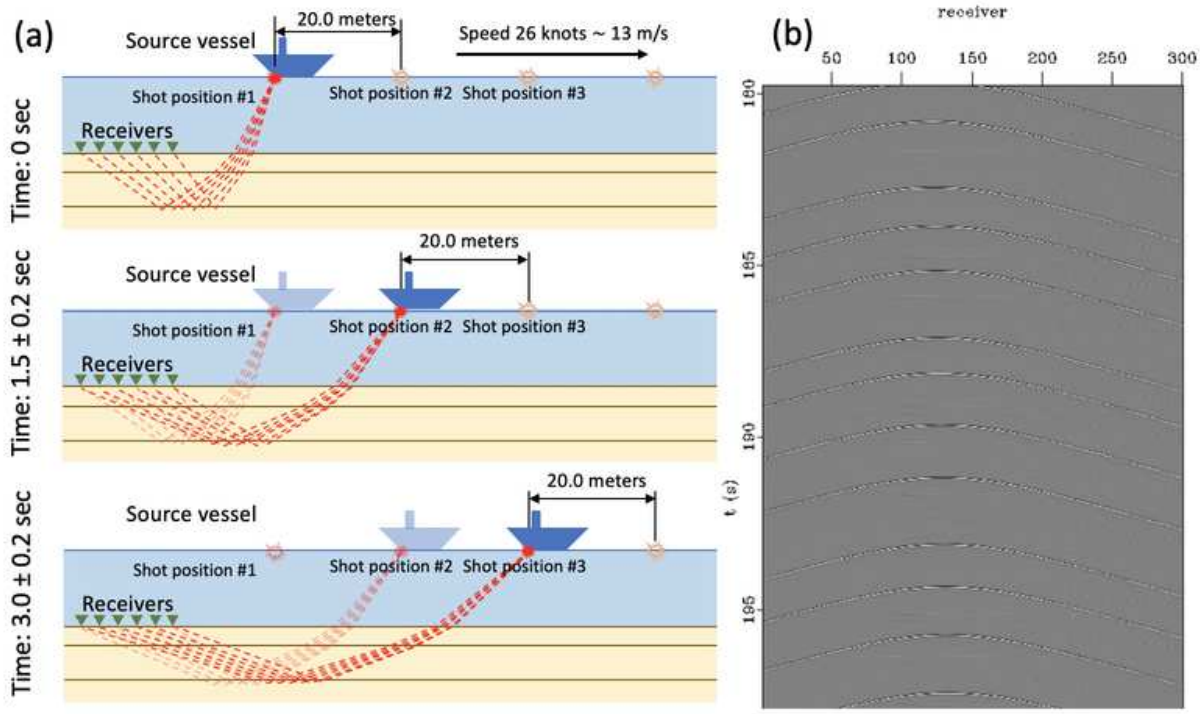

Figure 4.9: Simulated Blended acquisition geometry (a), and part of a continuous record with 12 shots (b).

- TFDN filter with a 2D window of 21 time samples x 11 spatial samples;
- Hard-thresholding with exponential decrease ($\tau_h = A \times 0.9^i$), where A is the input data maximum amplitude, and i is the current iteration.

Tests were conducted on common receiver gathers and then extended to the entire dataset (yet deblending in the receiver domain). The final results in the shot domain were obtained by sorting the data after deblending all receiver gathers.

Figure 4.10 shows the iterative deblending result on one receiver gather. The reconstructed[39] blended data and intermediate results for iterations 10 and 20 are shown in Figure 4.10b, c, and d. From those figures, it is clear that the blending noise is reduced after each iteration of the iterative approach. Analyzing the fully deblended data (Figure 4.10e), it is possible to observe how effective this deblending method is when compared with the original

[39]blended data after applying the pseudo-inverse operator

unblended data (Figure 4.10a). The methods effectiveness becomes more evident when we evaluate the deblending error[40] (Figure 4.10f), in which only a low amplitude residual noise can be observed.

As described in Section 3.4.2, due to the random characteristics of the blending noise in receiver gathers, it spreads over all the wavenumbers. Therefore, data analysis and filtering in the F-K domain are beneficial. Figure 4.11 shows the F-K spectra of one receiver gather at different steps of the iterative solution. When we compare the reconstructed blended data (Figure 4.11b) and the original unblended data (Figure 4.11a), we can observe the blending noise behavior. Applying the iterative deblending, after 10 iterations (Figure 4.11c), part of the noise was attenuated, and after 30 iterations (Figure 4.11d), only a residual blending noise can be seen.

Figure 4.12 shows the same steps illustrated in Figure 4.10, but now in the shot domain. As we can see, the results in the shot domain are comparable with those in the receiver domain when events from nearby sources are properly attenuated. Only a fraction of the blending noise still contaminates the data. Figure 4.12f illustrates the part of the signal that was not correctly deblended.

As the unblended datasets, the data that went through the iterative deblending were migrated using an RTM algorithm. The exact velocity and density models were used. Thus, I did not consider possible uncertainties associated with the models in the analysis.

Figure 4.13 compares the *baseline* migrated section for the unblended, blended and iteratively deblended data. The figure also shows the deblending error in the image domain. From the figure, we can see that the difference between the migrated sections (Figure 4.13a and Figure 4.13c) is subtle, which is confirmed by the small residual found in the error section (Figure 4.13d). A quantitative analysis, using the RMS amplitude extracted from the unblended and deblended datasets, will be presented in Section 4.3.3.

[40] difference between the unblended and deblended data

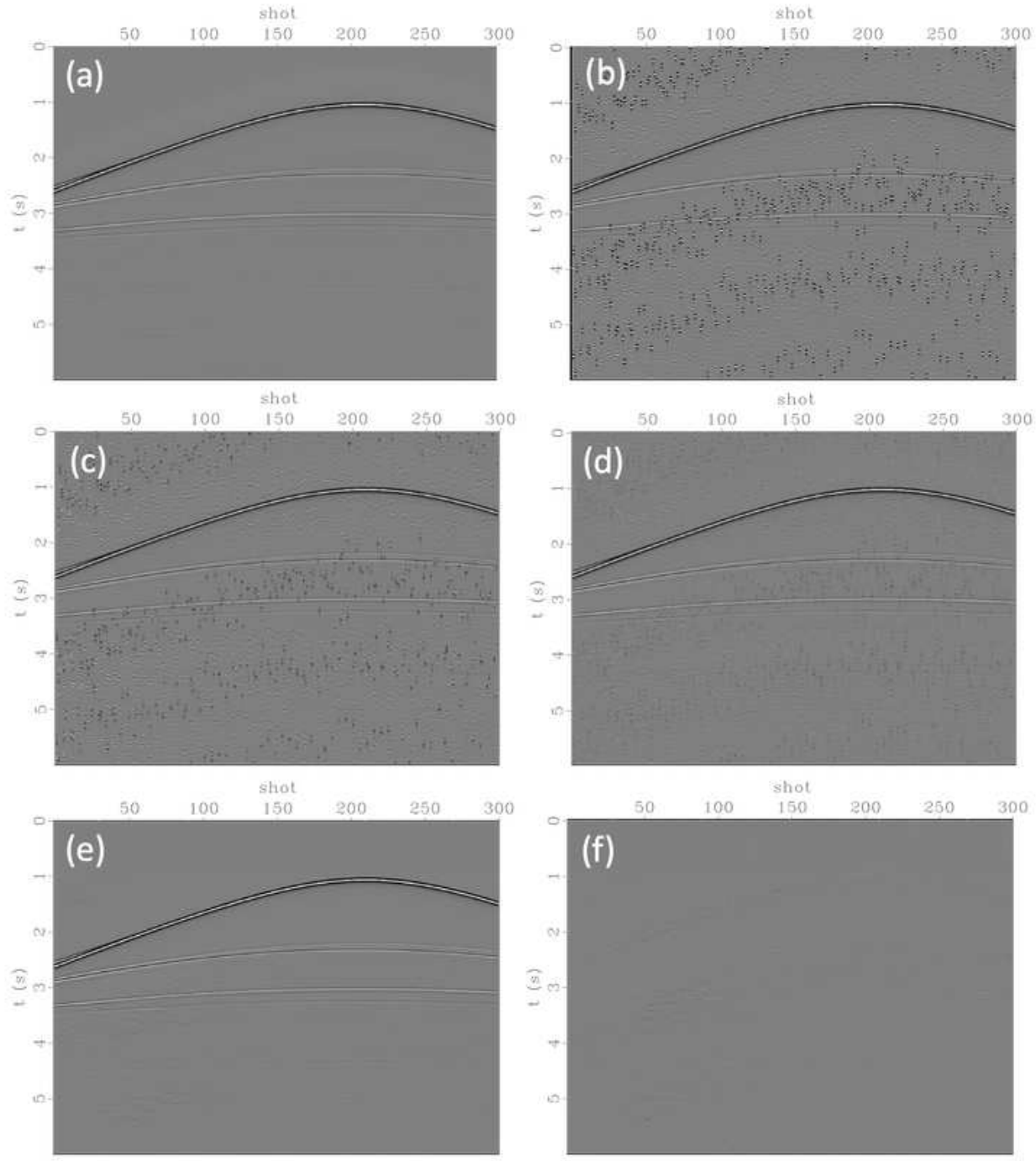

Figure 4.10: Iterative deblending results in the common receiver domain. Original unblended data (a), blended data (b), deblended data after 10 iterations (c), deblended data after 20 iterations (d), final deblended data (e), and deblending error (f). All plots have amplitude clipping to highlight the noise.

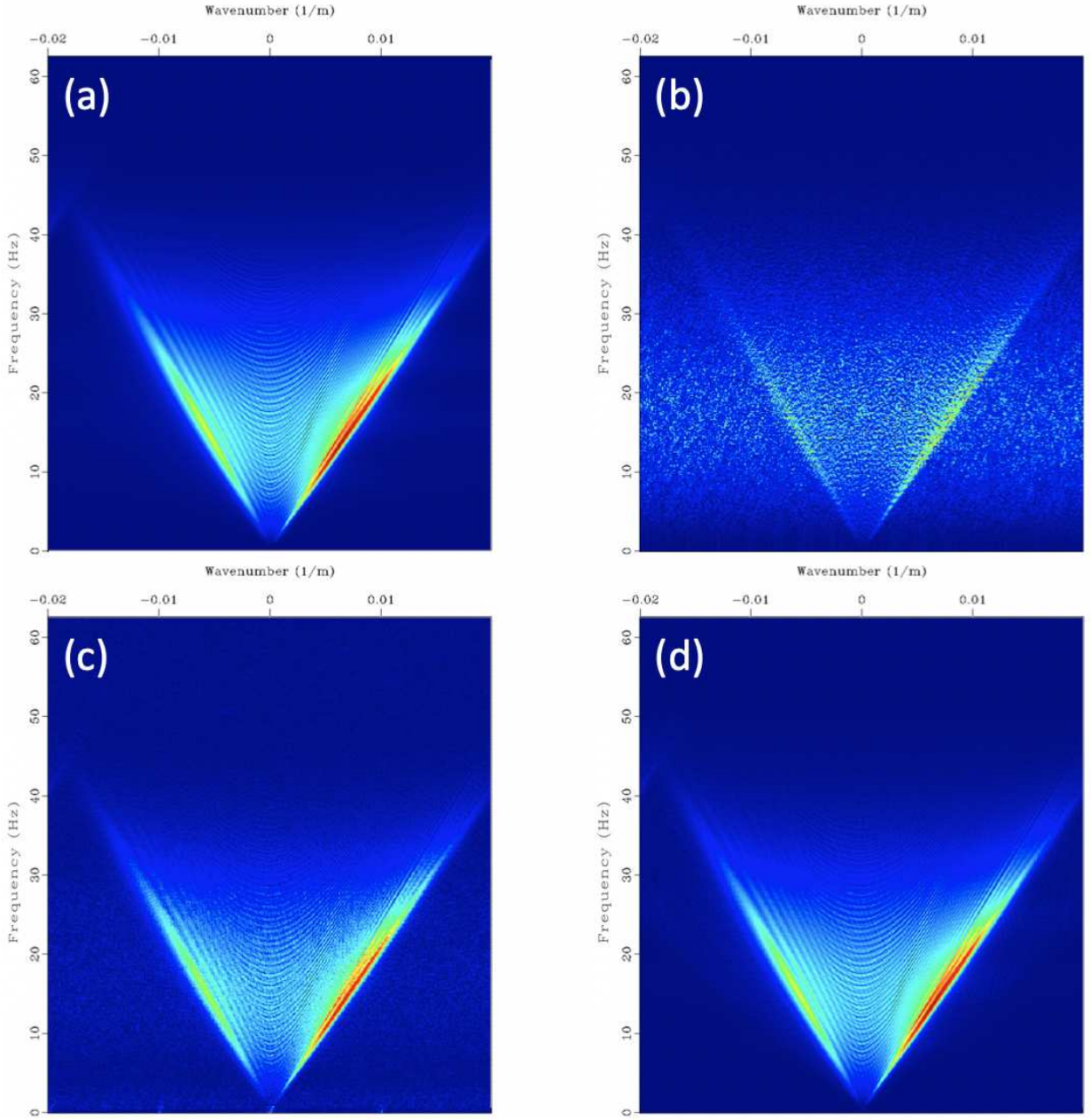

Figure 4.11: Iterative deblending results in the common receiver domain (F-K spectra). Original unblended data (a), blended data (b), deblended data after 10 iterations (c), and final deblended data (d).

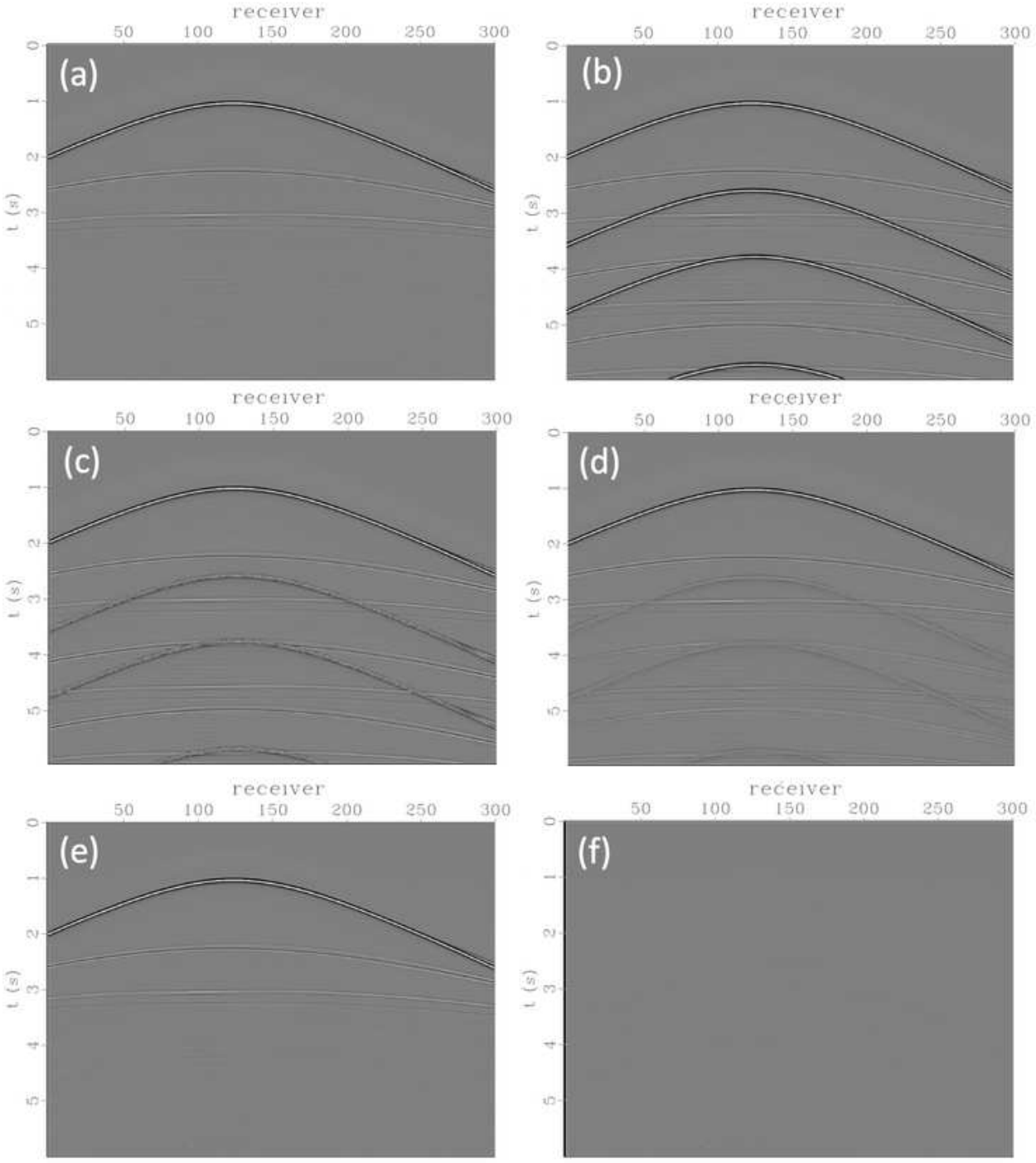

Figure 4.12: Iterative deblending results in the common shot domain. Original unblended data (a), blended data (b), deblended data after 10 iterations (c), deblended data after 20 iterations (d), final deblended data (e), and deblending error (f). All plots have amplitude clipping to highlight the noise.

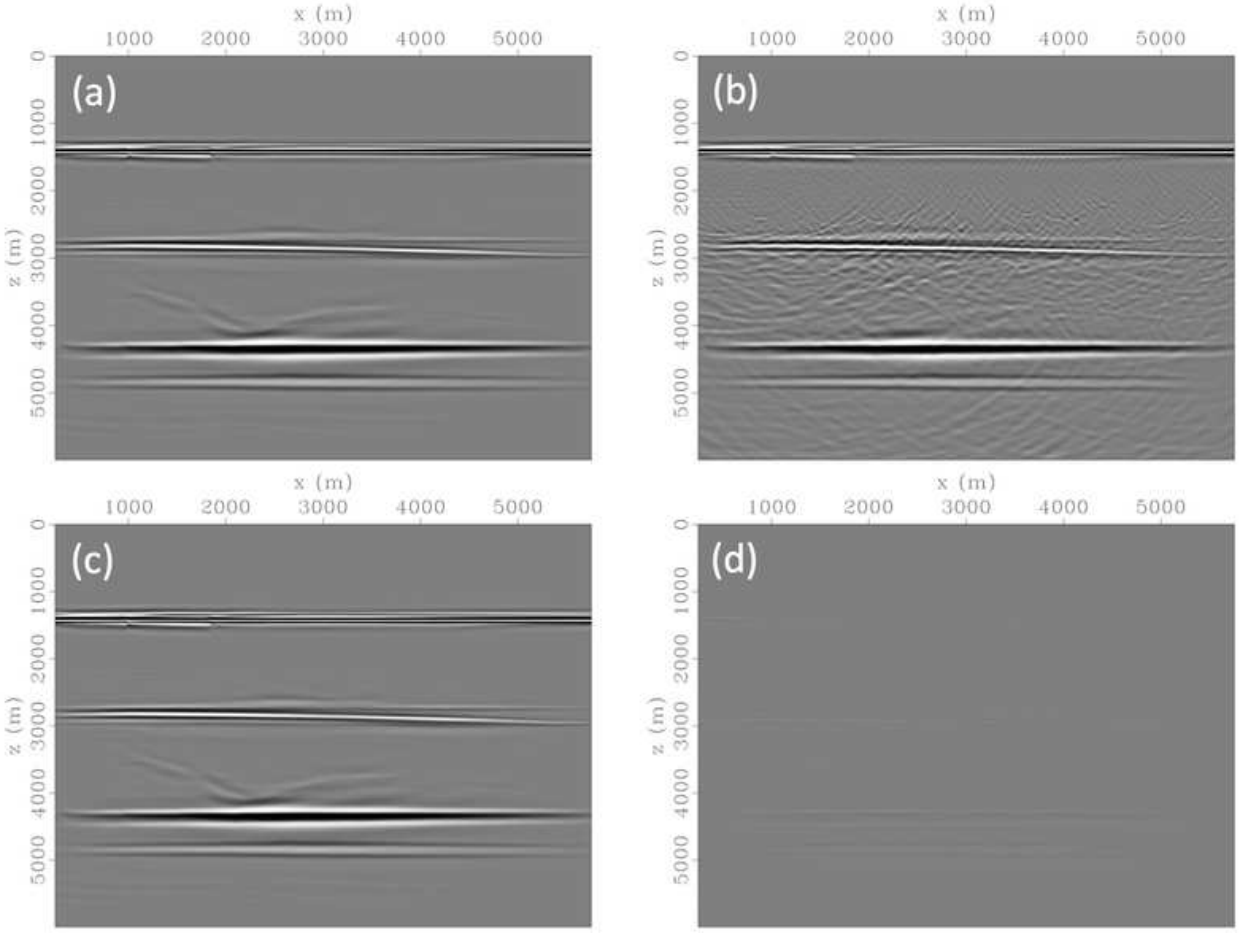

Figure 4.13: Migrated section for the unblended data (a), blended data (b), and iteratively deblended data (c). Deblending error in the image domain (d). All plots have amplitude clipping to highlight the noise.

4.3.2 Deblending by Compressive Sensing

As detailed in Section 3.5, methods based on compressive sensing use the prior knowledge that data is sparse in some domain. Therefore, the quality of the results depends directly on the sparse transformation used. Before running the tests, I investigated how sparse the data is in different domains to choose the appropriate transform to be used.

As mentioned in Section 3.5.4, the analysis consists of transforming the data into a sparse domain, where it is later reconstructed, considering only the k-largest coefficients. The relationship between the energy of the input and the reconstructed data is then calculated using equation 3.27. The objective of this analysis is to find the minimum number of coefficients necessary to reconstruct the signal with 99.9% of accuracy. Figure 4.14 illustrates the reconstruction analysis conducted in the Fourier, Wavelet, and Curvelet domains, considering all the data (3D transforms), or only one receiver (2D transforms).

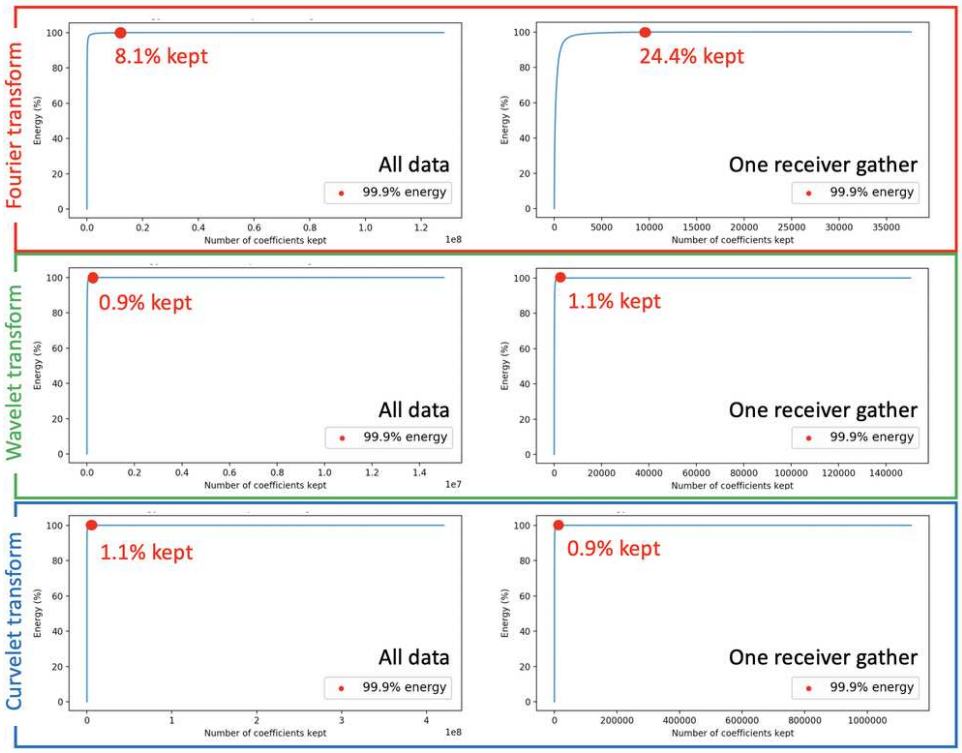

Figure 4.14: Synthetic data sparsity analysis in Fourier, Wavelet and Curvelet domains, considering one receiver gather or the entire data.

From the plots, it is possible to conclude that using curvelets or wavelets, only 1% of the coefficients are necessary to reconstruct the data with the target accuracy. In comparison, more than 8% are needed with the Fourier transform. This is an expression of data sparsity is the data when curvelets and wavelets are used as basis for data representation. Because of this, when the data is represented in this domain, the probability of finding appropriate solutions to an inverse problem where we impose sparsity is greater.

Considering the sparse analysis results, I have performed tests with the three transforms (Fourier, Wavelet, and Curvelet transforms). For all tests, I used the SPGℓ_1 solver with a misfit limit of $\sigma = 0.01$ to find the solution for equation 3.21 (repeated bellow):

$$\min_{\mathbf{f}} \|\mathbf{f}\|_1 \text{ subject to } \left\|\mathbf{b} - \mathbf{T}\mathbf{S}^H\mathbf{f}\right\|_2 \leq \sigma.$$

Figure 4.15 shows the results for the *Fourier transform*, where 207 iterations of the solver were required for the solution to converge. The *Wavelet transform* (Figure 4.16) required 400 iterations, and finally, the *Curvelet transform* (Figure 4.17) required an average of 250 iterations per receiver for the solution to converge.

Comparing the results for the three transforms, the Fourier and Curvelet results are very similar, with excellent signal recovery and little residual noise. The Wavelet transform delivered results with more artifacts, which is something that needs to be better investigated[41]. The results lead us to conclude that although the Fourier transform is less sparse, it is sparse enough for a good recovery on synthetic data without noise.

Considering the conclusions obtained with noise-free data, I repeated the previous test with the Fourier transform, but now adding 5% *of random noise* into the *blended data*. Figure 4.18 shows the results for the noisy data, where 114 iterations were required for the solution to converge. As we can see, deblending data with random noise generates results of similar quality as obtained previously. In fact, the deblending technique, besides attenuating the blending noise, also mitigated the introduced random noise.

[41] the tests were performed with real and complex wavelets, and in both cases the results were similar. Some works suggest that the use of complex wavelets generate superior results, although I have not been able to confirm this statement.

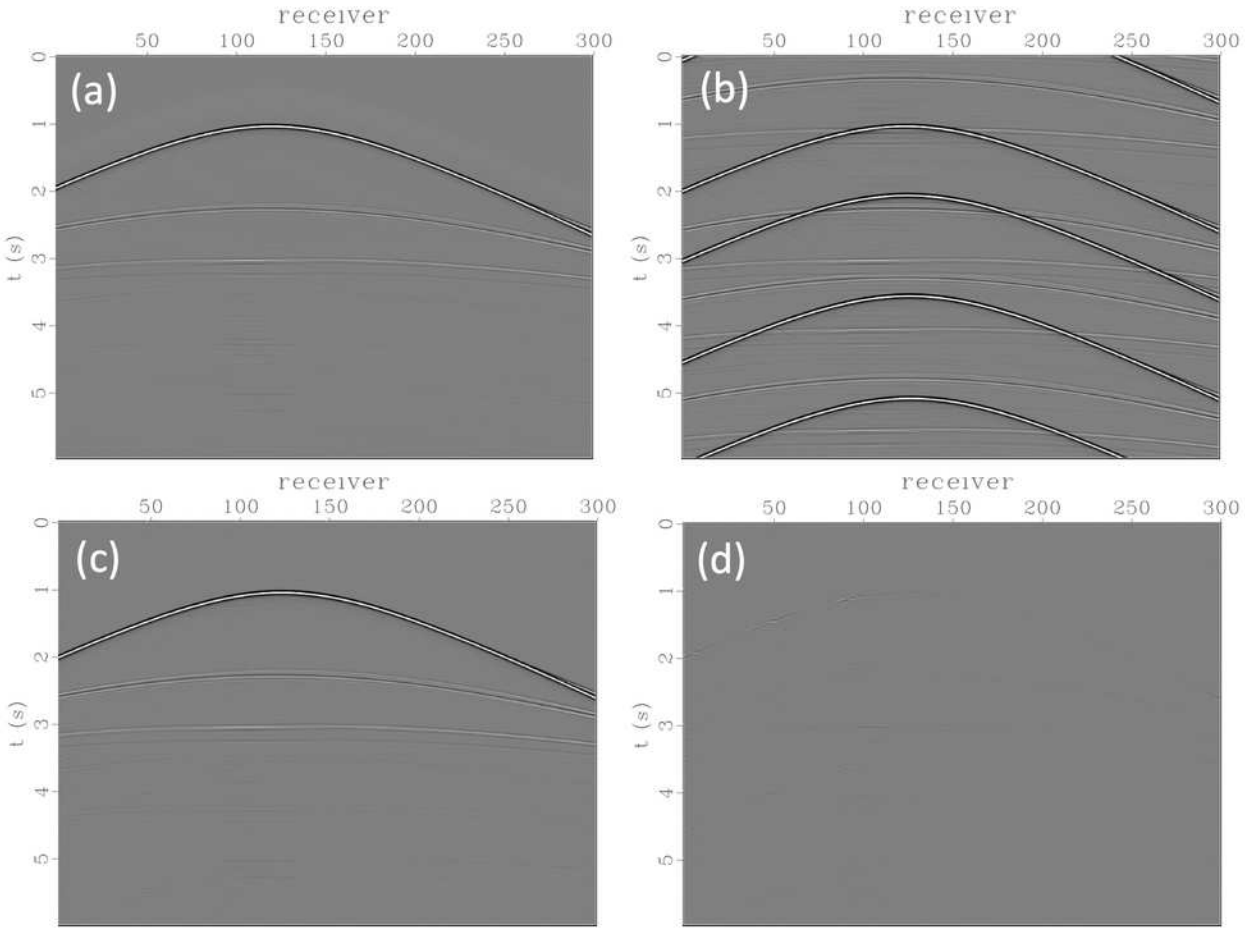

Figure 4.15: Deblending by compressive sensing results using the **Fourier transform** (common shot domain). Original unblended data (a), blended data (b), deblended data (c), and deblending error (d). All plots have amplitude clipping to highlight the noise.

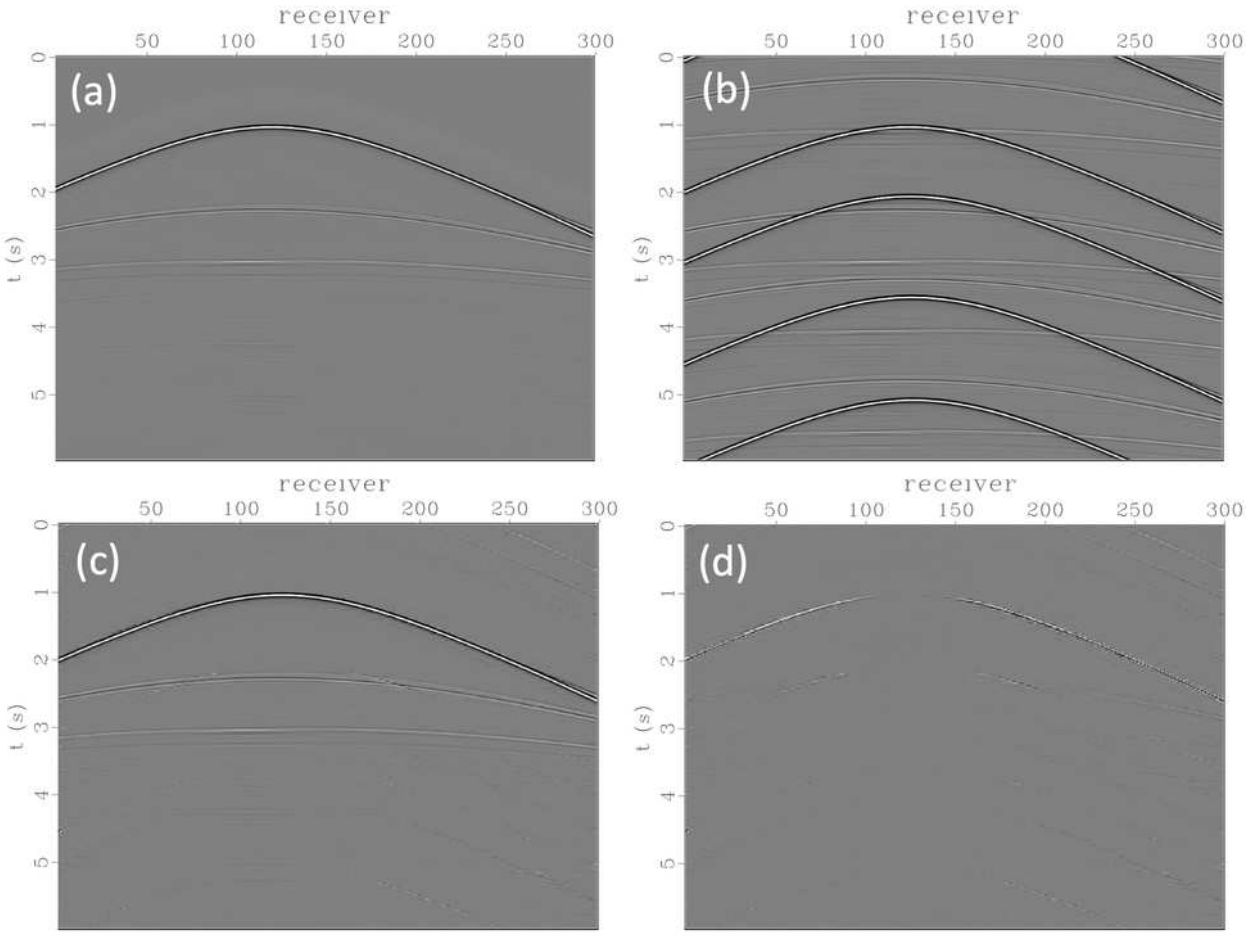

Figure 4.16: Deblending by compressive sensing results using the **Wavelet transform** (common shot domain). Original unblended data (a), blended data (b), deblended data (c), and deblending error (d). All plots have amplitude clipping to highlight the noise.

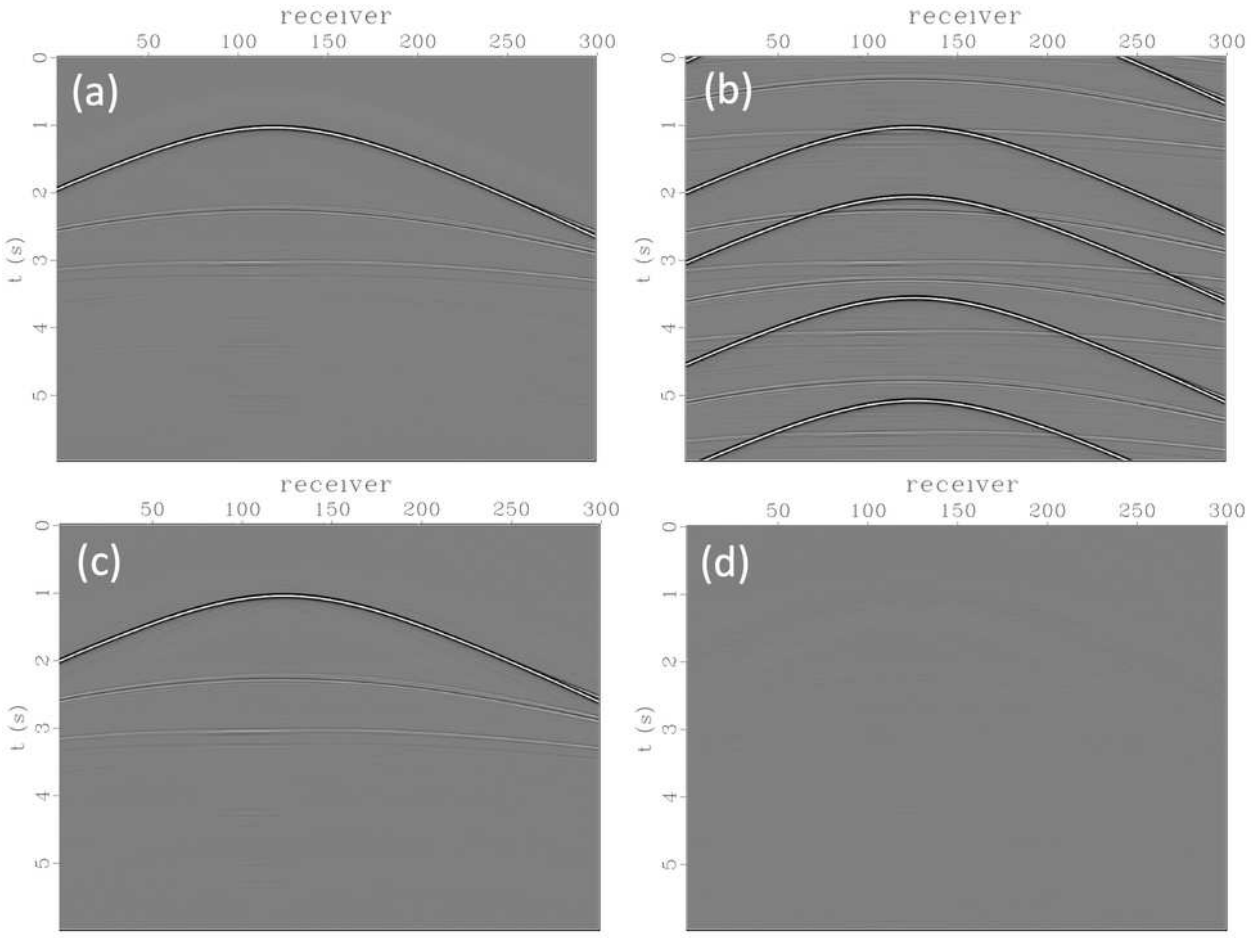

Figure 4.17: Deblending by compressive sensing results using the **Curvelet transform** (common shot domain). Original unblended data (a), blended data (b), deblended data (c), and deblending error (d). All plots have amplitude clipping to highlight the noise.

Figure 4.19 shows the deblended data with and without noise, and compares it with the blended data with noise (the amplitudes were clipped to highlight the noise). The attenuation of random noise is directly associated with the process of inducing sparsity into the solution.

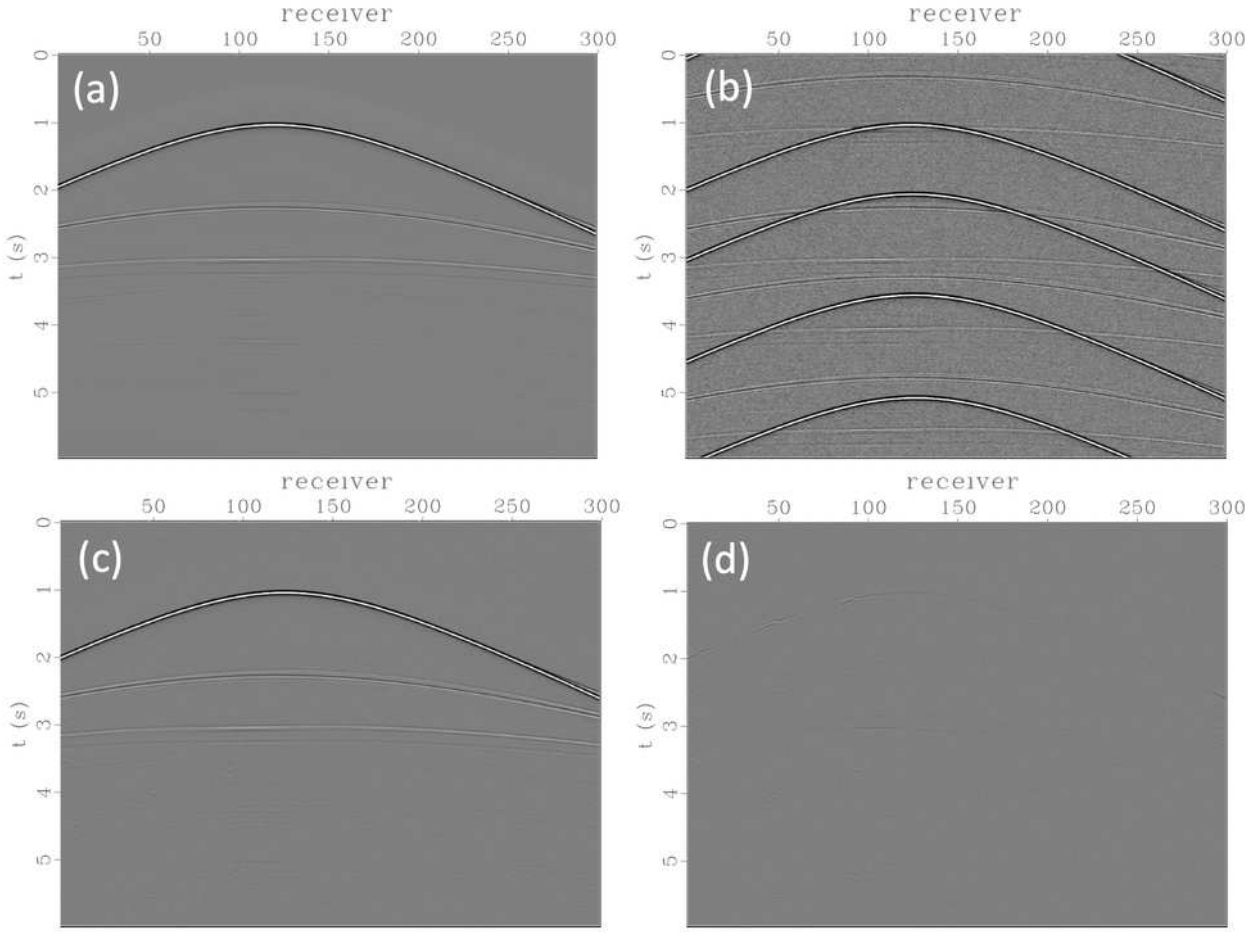

Figure 4.18: Deblending by compressive sensing results using the **Fourier transform** with **noisy data** (common shot domain). Original unblended data (a), blended data with 5% of random noise (b), deblended data (c), and deblending error (d). All plots have amplitude clipping to highlight the noise.

Similar to what was done with the iterative deblending, the data that went through the deblending by compressive sensing was migrated using an RTM algorithm. Figure 4.20 compares the unblended, blended and CS deblended data (based on Fourier and Curvelet transforms). It also includes the deblending error. From the migrated sections of the deblended data (Figure 4.13c, Figure 4.20c and Figure 4.20e), I observe that both methods are

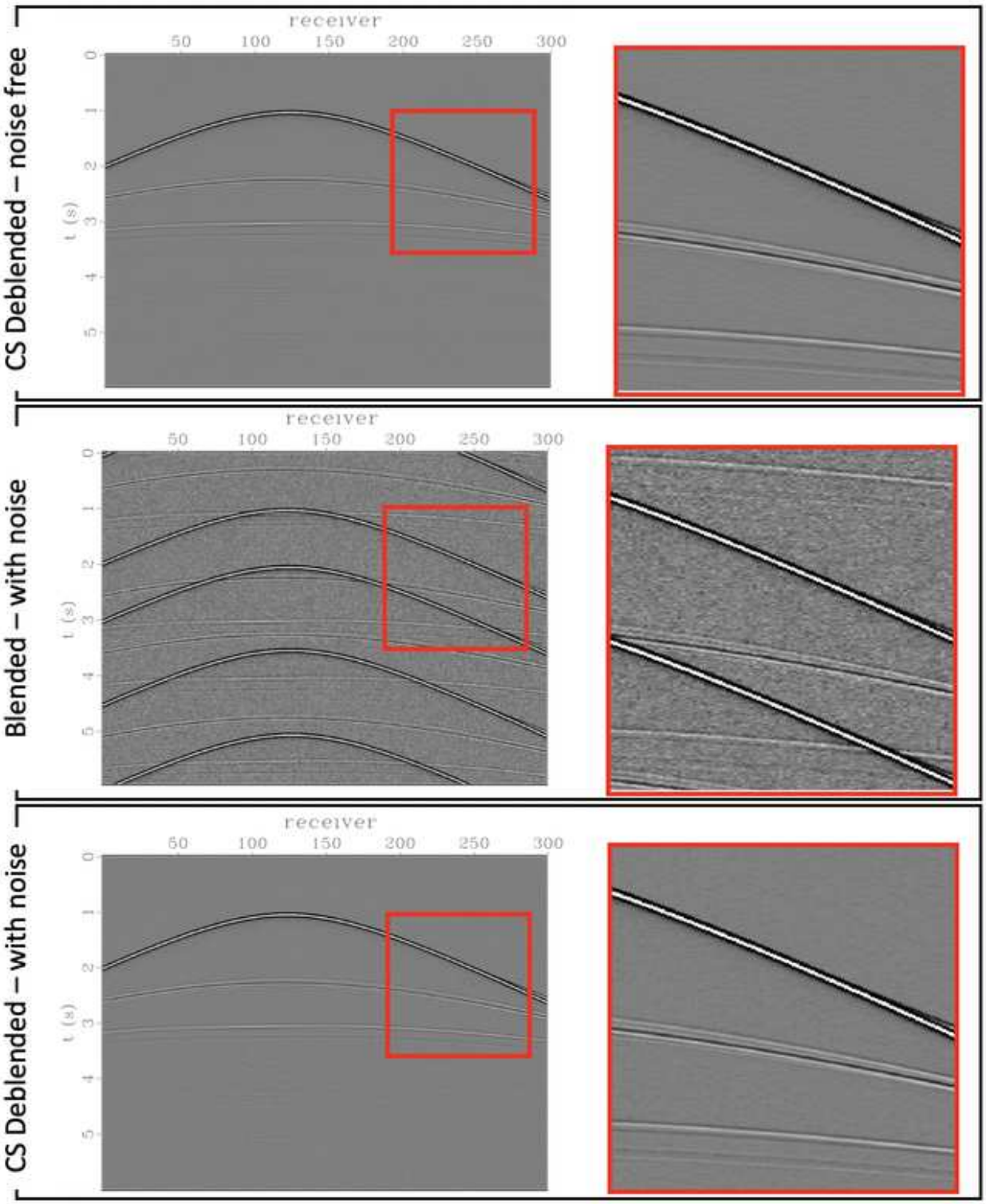

Figure 4.19: Comparative of the deblending by compressive sensing with and without random noise. The random noise was introduced in the blended data before the deblending process. All plots have amplitude clipping to highlight the noise.

equally efficient in minimizing the effects of blending noise, generating images with similar quality. Comparing now only the results of the CS deblending (with Fourier and Curvelet as sparse transform), we observe that the Curvelet transform generates results with fewer artifacts, and with more consistent amplitudes (Figure 4.20d and Figure 4.20f).

Table 4.2 compares the deblending methods in terms of overall quality, execution time, difficulty in parameterization, and the need for computational resources. The same workstation was used to perform all tests. In summary, the results obtained are similar in terms of quality, with deblending by compressive sensing generating slightly superior results. However, this small gain in quality requires at least 10x more computational resources, and when used with the curvelet transform, it takes 2x more time.

Table 4.2: Deblending methods comparative analysis on synthetic data

Method	**Iterative**	**CS (Fourier)**	**CS (Wavelet)**	**CS (Curvelet)**
Results quality	High, with minor residual noise	High, with minor residual noise	Medium, with visible artifacts	High, with minor residual noise
Execution time	20h	9h	26h	42h
Memory requirement	Low (less than 2 Gb of RAM)	Medium (around 16 Gb of RAM)	Medium (around 16 Gb of RAM)	High (more than 32 Gb of RAM)
Parameterization	Hard (Coherence filter parameters)	Easy (solver parameters)	Easy (solver parameters)	Easy (solver parameters)

4.3.3 Quantitative Analysis

As the results presented in the previous sections are very similar, a more quantitative analysis becomes necessary. In this section, I show a comparison, in terms of the RMS amplitude, between the unblended and deblended datasets (without noise). At this point, only the baseline data is being analyzed. The 4D analysis will be detailed in Section 4.4.

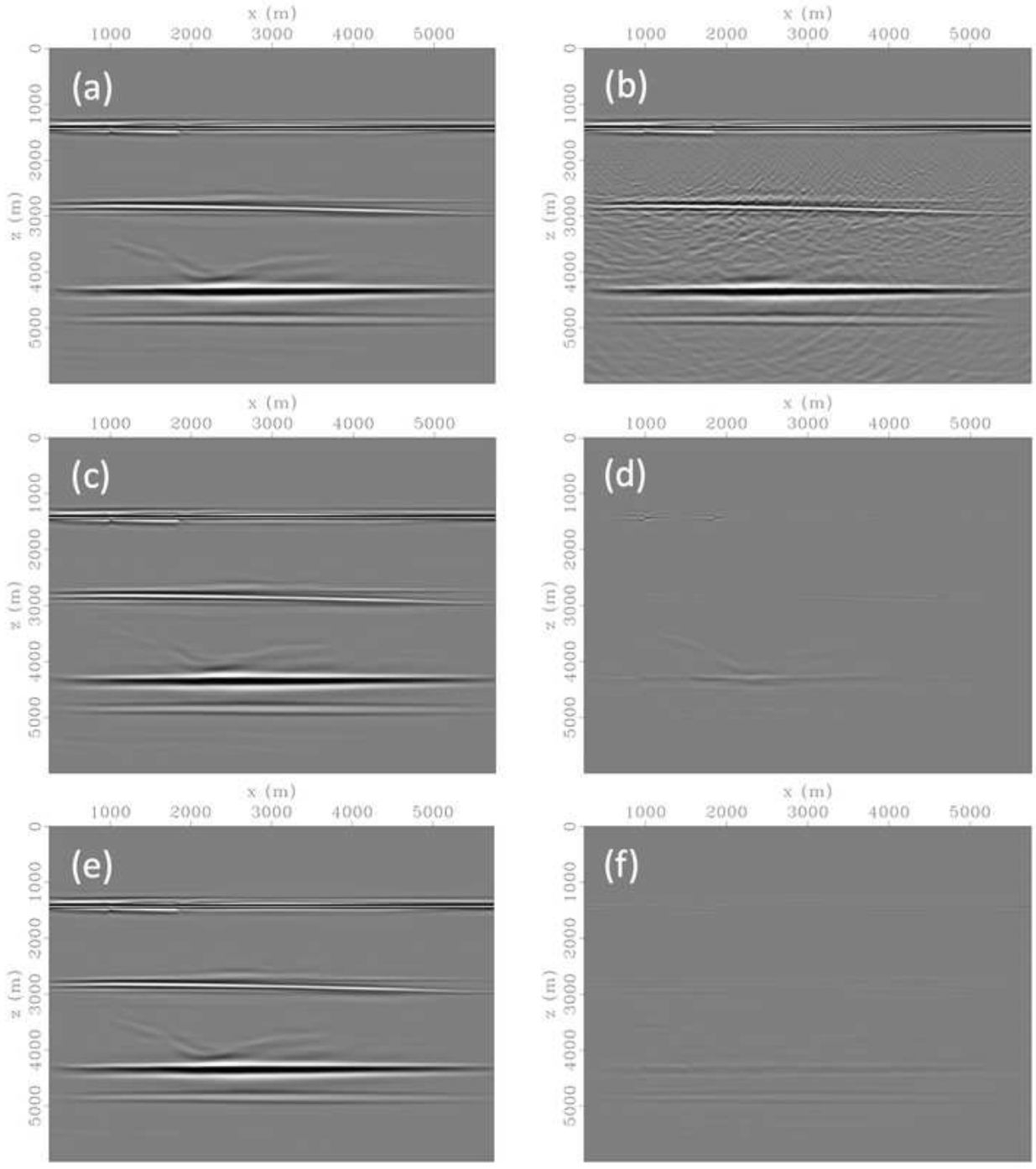

Figure 4.20: Migrated section for the unblended data (a), blended data (b), CS deblended data with **Fourier)** as sparse transform (c), and **Curvelet** as sparse transform (e). **Deblending error** in the image domain with **Fourier** as sparse transform (d) and **Curvelet** as sparse transform (f). All plots have amplitude clipping to highlight the noise.

Since the image domain is the natural domain for the extraction of attributes, I extracted the RMS amplitudes on that domain, which means on migrated sections. For that, I have chosen two reflectors: the reservoir interval and a pre-salt layer. Around each one of those reflectors, a window was defined, and then the RMS amplitude was extracted using equation 4.2, generating a RMS curve for each dataset (unblended, iteratively deblended, and CS deblended with Curvelet[42]). Figure 4.21 compares the RMS curves for the reservoir interval, and Figure 4.22 for the pre-salt layer.

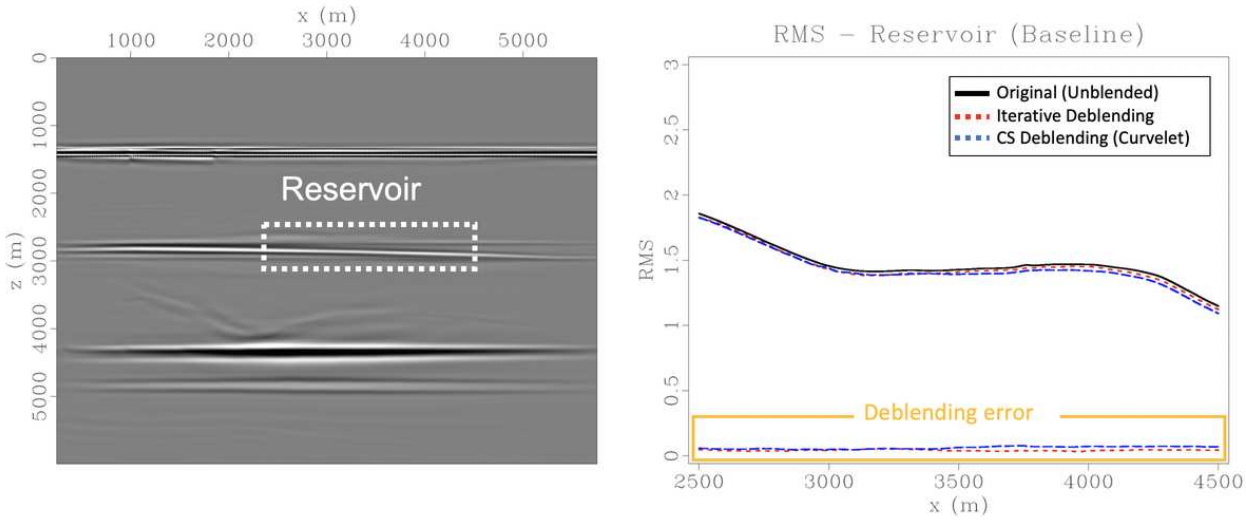

Figure 4.21: RMS amplitude extracted from baseline data around the **reservoir** interval (white dashed line). The black line represents the RMS amplitude from the unblended data, the red line represents the RMS amplitude from the iteratively deblended data, and the blue line represents the RMS amplitude from the CS deblended data with Curvelet. The deblending errors are represented at the bottom of the plot.

From the figures, one observes that the deblending methods generate results compatible with the original unblended data. In both cases, the errors are small and laterally consistent, confirming that for the synthetic data, the deblending methods are able to preserve the amplitude variations present in the data.

After analyzing the baseline, I repeated all tests with the monitor data, getting results consistent with those from the baseline data. Because of that, I will not present these results.

[42]I chose Curvelet because it generated more consistent results in the image domain

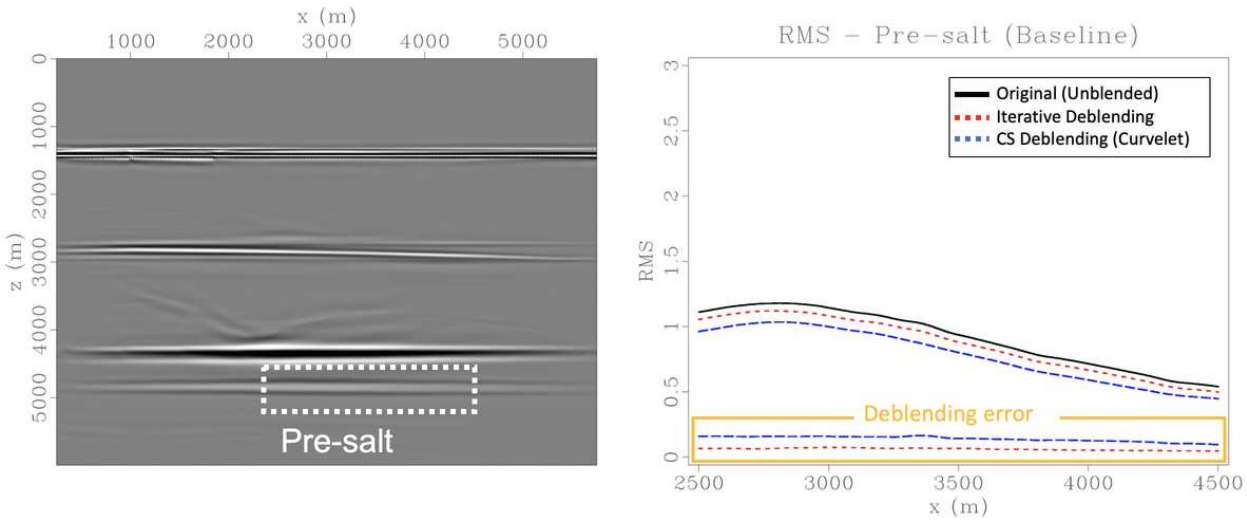

Figure 4.22: RMS amplitude extracted from baseline data around the **pre-salt** layer (white dashed line). The black line represents the RMS amplitude from the unblended data, the red line represents the RMS amplitude from the iteratively deblended data, and the blue line represents the RMS amplitude from the CS deblended data with Curvelet. The deblending errors are represented at the bottom of the plot.

4.4 Time-lapse analysis

In the previous sections, the baseline data were analyzed from the perspective of deblending methods. Additionally, although not presented, the results obtained for the monitor data were similar to the baseline in terms of efficiency and residual noise level. In this section, I compare the deblended baseline and monitor data in terms of the NRMS amplitude. This analysis is essential to measure the effect of blended acquisition and deblending techniques on 4D projects.

The first step is to evaluate the 4D difference between the baseline and monitor data after applying each deblending method. Figure 4.23 shows the 4D difference for the unblended, iteratively deblended, and CS deblended data with Curvelet transform. Regardless of the deblending method used, the 4D difference sections are very similar. In fact, it is not possible to distinguish the results based only on visual comparison. This result reinforces the consistency and effectiveness of those methods, when compared with the original unblended data.

For all tested methods, the amplitude difference in the reservoir interval is considerable, and outside the reservoir is very small. As explained in Section 4.1, changes in reservoir properties affect not only the reflection coefficients, but also the transmission coefficients. Additionally, according to modeling carried out in the area, time-shifts also occur, but they are not relevant. This means small 4D differences in reflectors below the reservoir are expected.

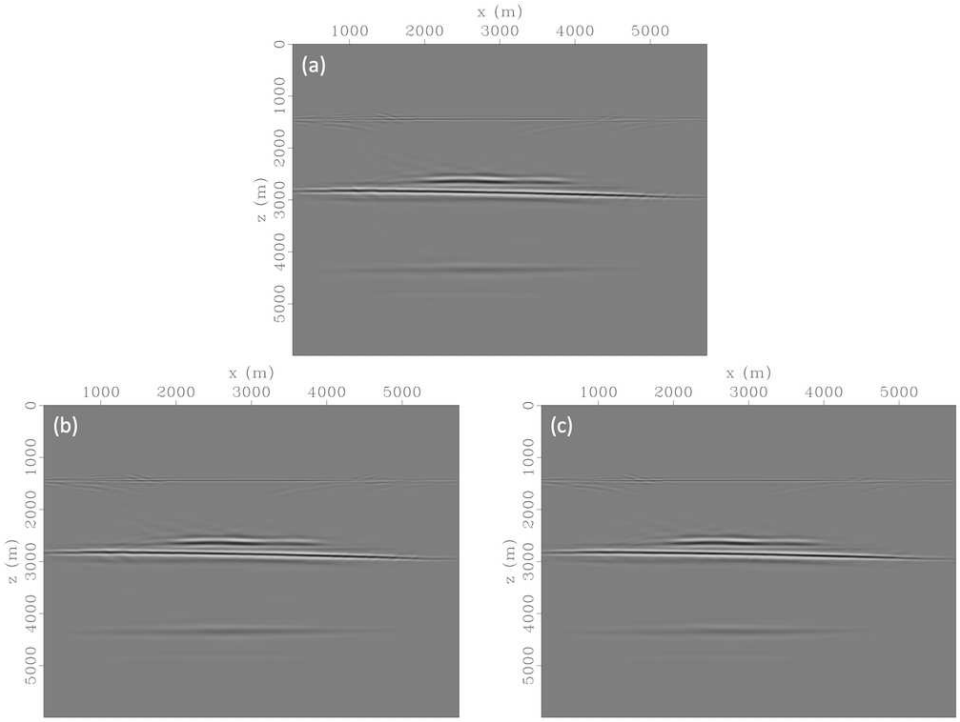

Figure 4.23: 4D difference for the unblended (a), iteratively deblended (b), and CS deblended data (c) with Curvelet transform. All plots have amplitude clipping, and were displayed at the same gain.

Since visual inspection is not enough to attest the quality of the results, the NRMS amplitude has been extracted. For this, I have applied equation 4.1 to the results obtained. For 2D seismic data, an NRMS curve is generated. For 3D data, an NRMS horizon would be obtained. Figure 4.24 shows the NRMS curve for the iteratively deblended data, and Figure 4.25 for the CS deblended data with Curvelet transform.

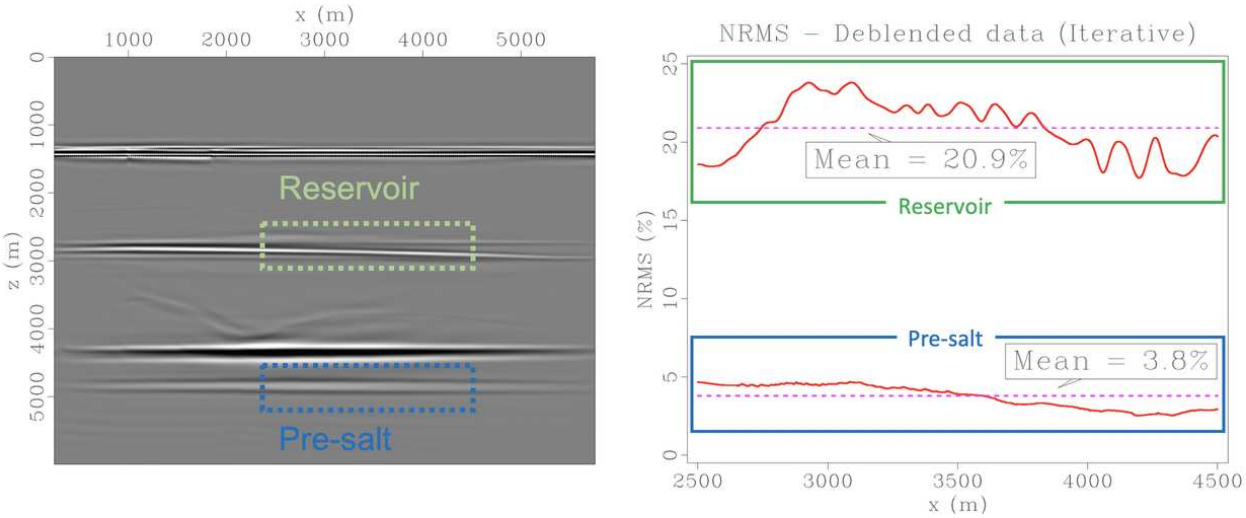

Figure 4.24: NRMS amplitude extracted from the iteratively deblended data. The red line is the NRMS amplitude, and the pink dashed line is the mean value.

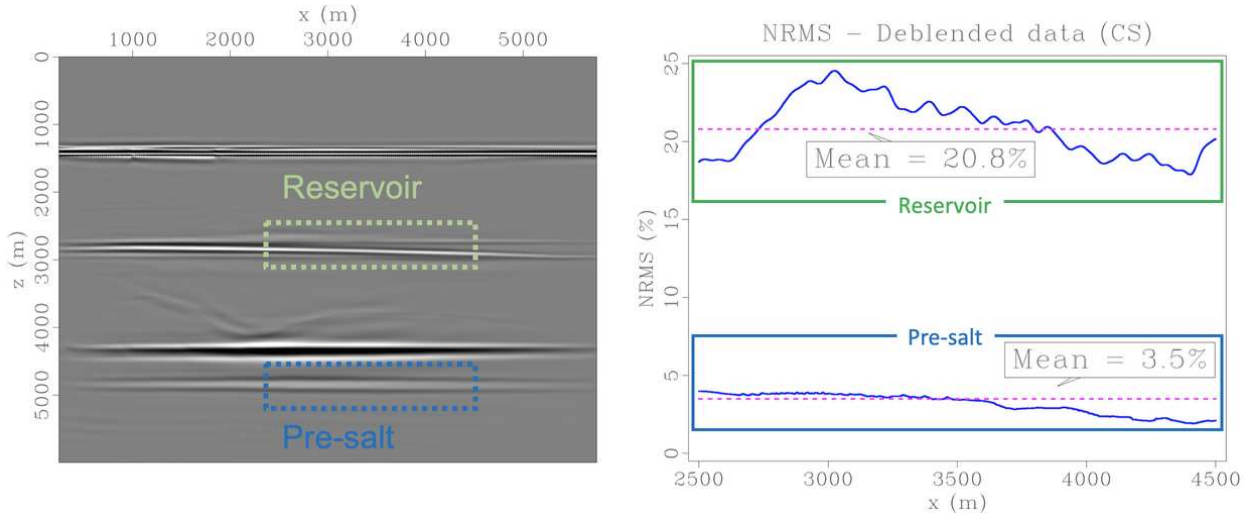

Figure 4.25: NRMS amplitude extracted from the CS deblended data with Curvelet transform. The blue line is the NRMS amplitude, and the pink dashed line is the mean value.

As we can see, the average value of the NRMS amplitude in the reservoir interval and in the pre-salt layer, for each deblending method, are similar to those obtained with the original data (Figure 4.8). Table 4.3 summarizes those values and Figure 4.26 shows all NRMS curves in the same plot to simplify the comparison.

Table 4.3: Mean NRMS comparative.

Mean NRMS	**Unblended (Reference)**	**Iterative**	**CS (Curvelet)**
Reservoir interval	20.5%	20.9%	20.8%
Pre-salt layer	3.3%	3.8%	3.5%

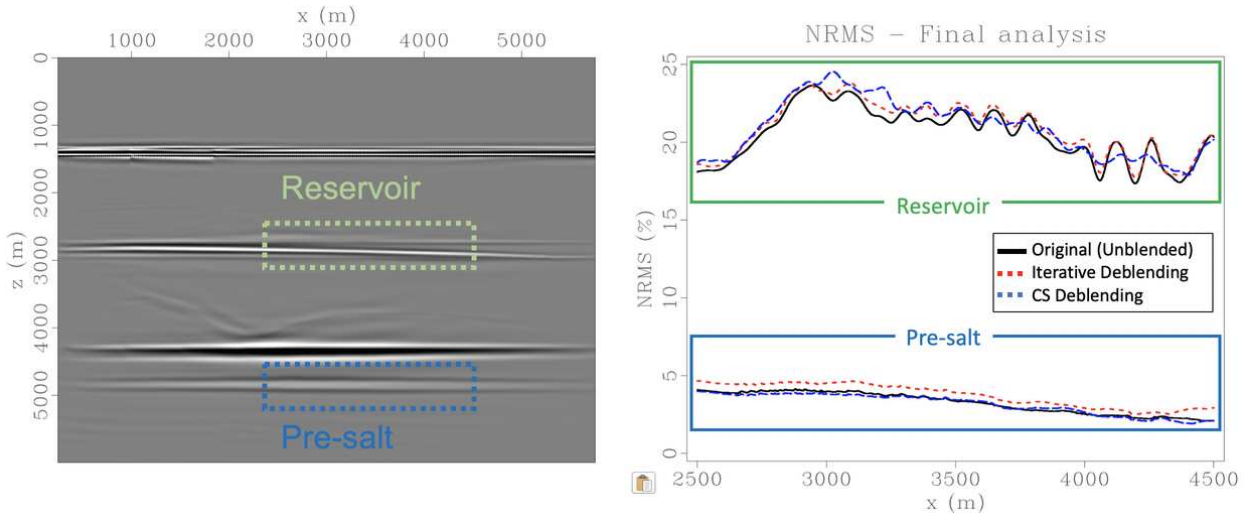

Figure 4.26: Comparative analysis of the NRMS amplitude extracted from different datasets. The black, red, and blue lines are the NRMS amplitude obtained from the unblended, iteratively deblended, and CS deblended datasets, respectively.

Considering the original unblended data as a reference, *the NRMS values introduced by the deblending methods are between* ***0.2%*** *and* ***0.5%***, which is very small. This is an important quantity because it measures the impact of a blended acquisition in a 4D project. Since our experiment did not include any other external effects, this value represents only the differences generated by the blended geometry in our acquisition and by the deblended procedures applied.

4.5 Synthetic Data Analysis - Final Remarks

The results presented in this chapter, with synthetic data, demonstrate the effectiveness of deblending techniques in recovering deblended data. Such as any seismic processing technique, these methods are not capable of generating a completely noise-free solution, even on synthetic data. However, the remaining noise/artifacts cause marginal effects in the final image, as was demonstrated.

In terms of quality, both methods results are very similar, eliminating most of the blending noise, with the CS deblending generating "cleaner" shot gathers than the iterative deblending. However, when the execution time and need for computational resources are taken into account, the iterative deblending appears as the best choice since it requires less powerful computers, and runs faster.

The 4D analysis provided valuable insight into how blending noise and deblending techniques affect 4D projects. The repeatability analysis, through the NRMS amplitude, demonstrated that the impact on the 4D signal is minimal, of the order of 0.5% for the analyzed data. Although the 4D analysis used data free of noise, as shown, the introduction of noise only affected the quality of the final image marginally.

Based on the obtained NRMS level, I consider 0.5% to be the minimum achievable NRMS value for our 4D blended project. As a reference, I compare these values with the NRMS value calculated for the seismic data from the Jubarte field, which was 4.5%. That is, opting to repeat the Jubarte seismic survey as a blended acquisition would increase the NRMS values for a time-lapse project to 5%, which is still a very low value.

CHAPTER 5

DEBLENDING APPLICATION - FIELD DATA

The final stage of validation of any geophysical technique is the application to field data. This chapter shows the use of iterative and compressive sensing deblending algorithms applied to a real seismic data acquired offshore Brazil.

5.1 Data Overview

In December of 2012, Petrobras, the Brazilian national oil company, installed the first Permanent Reservoir Monitoring (PRM) system offshore Brazil (Dariva *et al.*, 2016; Thedy *et al.*, 2013). The system, placed in the Jubarte Field (north of the Campos Basin), is composed of underwater cables with multicomponent optical receivers placed on the seabed at 1,200 meters depth. The system covered approximately 9 square kilometers with 712 receivers to acquire passive and active seismic during the life of the field.

From 2012 to 2015, three active source seismic surveys acquired high quality 4D data, of excellent repeatability, and good signal-to-noise ratio. Each survey was composed of a shot carpet covering an area of approximately 120 square kilometers. The shooting carpet consisted of more than 400 sail lines, spaced by 25 meters. The distance between each sail line was also 25 meters. Table 5.1 summarizes the main acquisition parameters.

Figure 5.1 shows the field location and the baseline survey post-plot, with the underwater cables in black and the shot locations in green (Dariva *et al.*, 2016).

The surveys were designed considering the non-interference assumption, which means the boat speed was properly set to accommodate a record length of 10 seconds without any overlap between shots. The blended data used in this chapter was numerically blended from individual shot gathers.

Table 5.1: Jubarte PRM system general acquisition parameters.

	Parameters
Shot Survey Area	11km x 11km
Shooting interval	25m
Number of sail lines	441
Sail line spacing	25m
Shots per sail-line	441
Orientation	129.3°/309.3°
Receiver patch	3km x 3 km
Receiver interval	50m
Number of receivers	712
Number of components	4 (1 H + 3G)
Record Length	10,240 ms
Trace density	3.8 million traces per km^2
Azimuthal coverage	Full-Azimuth
LC filter	1 Hz @ 6 dB/octave
HC filter	214 Hz @ 341 dB/octave

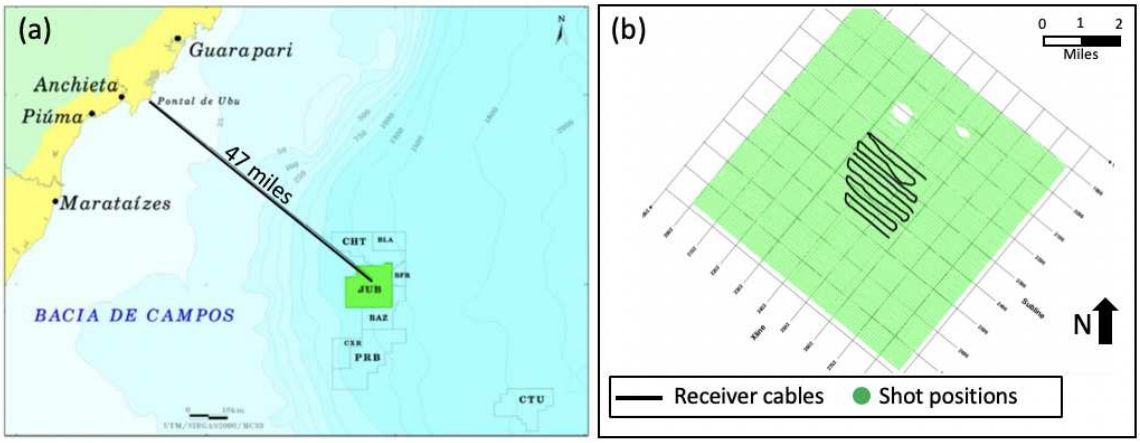

Figure 5.1: Jubarte field location (a), and PRM system geometry (b). Adapted from Dariva *et al.* (2016).

5.2 Blending Simulation

As mentioned in the previous section, the field data provided by Petrobras was acquired using a traditional geometry, without overlapping shots. Therefore, to use the data in this project, it was necessary to blend the shots numerically following the procedure described in Chapter 2.

The simulation was performed on a sail line composed of 440 shots, where each shot gather had 712 receivers. Figure 5.2 shows shot gathers for the hydrophone and accelerometer horizontal in-line components. Due to the computational resources available, only the first 300 shots and the first 300 receivers are used in the simulation. Also, the data were resampled from 2 to 4ms, and cut between 0 and 6 seconds. A high-cut filter was applied before resampling it, to prevent aliasing. Finally, since four components were available (hydrophone and 3C accelerometers), I used the following strategy:

- The pressure component (hydrophone) was used in most of the tests for the methods comparison;
- The accelerometer horizontal in-line component was used in the final validation to assure that the conclusions are extensible to 4C blended data.

As the main objective of this work is to simulate an operational gain in terms of acquisition time and on cost reduction, a source density ratio (SDR) of 4 was considered. This is equivalent to an increase of four times the boat speed (Section 1.6 presents more details about blended seismic acquisition performance indicators).

Applying the modeling described by equation 2.3 to the conventional field data, a continuous record of approximately 450 seconds was generated. The encoding of the shots was based on the random time-dithering scheme (Section 2.2) with shooting intervals of 1.5 ± 0.2 seconds. Figure 5.3 shows a 25 seconds section of the continuous record generated for hydrophone component.

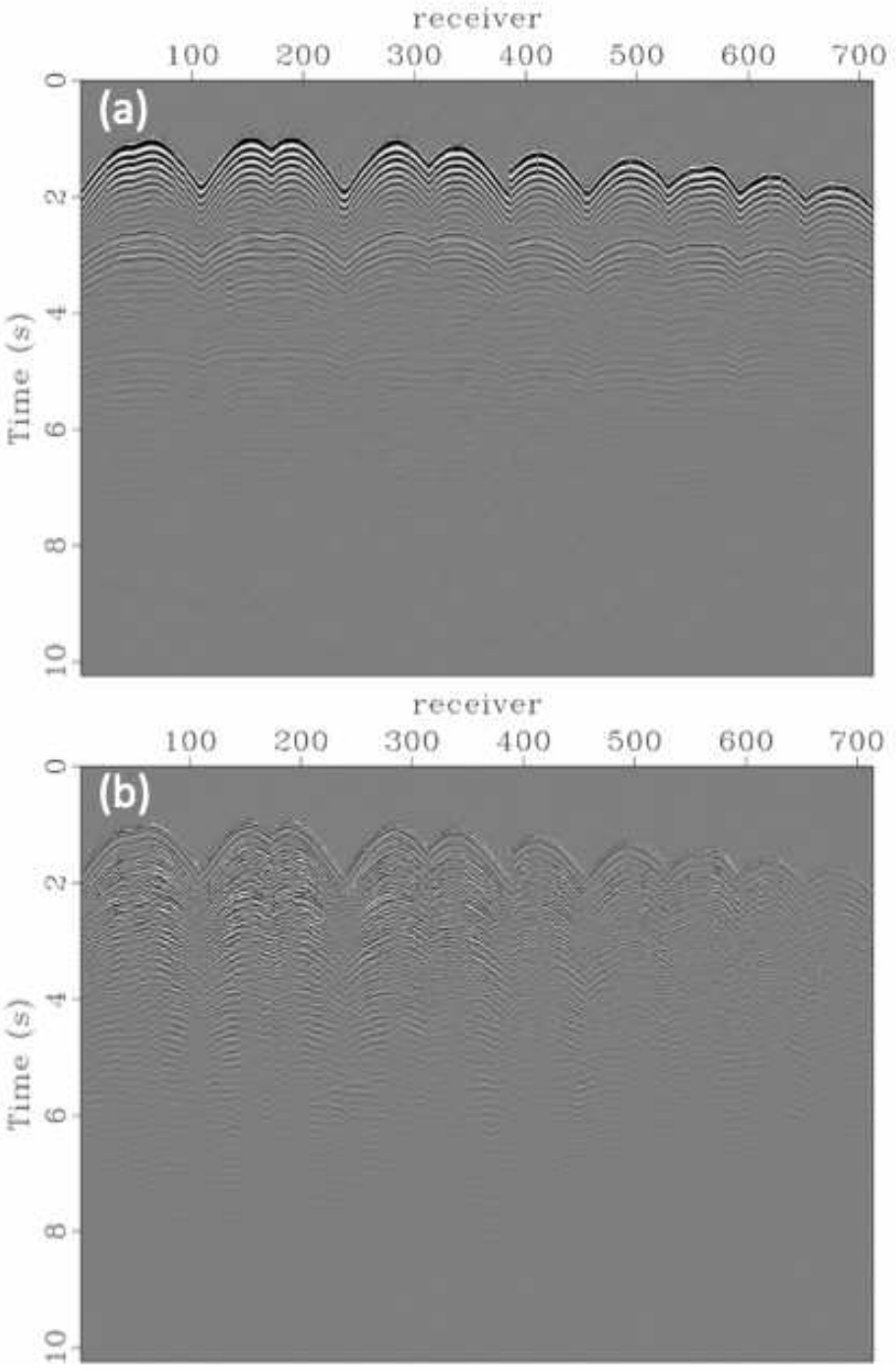

Figure 5.2: Raw shot gathers extracted from Jubarte seismic survey. Hydrophone component (a). Accelerometer horizontal in-line component (b).

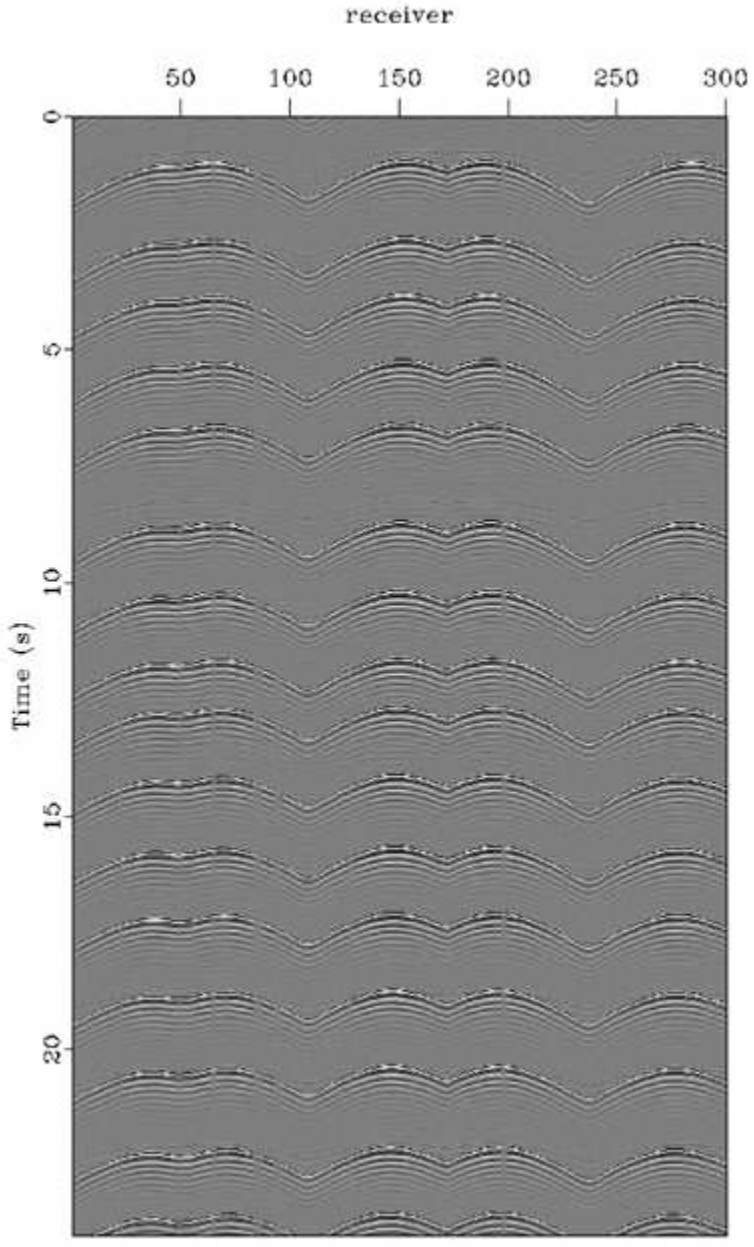

Figure 5.3: Continuous blended shot gather generated from the Jubarte seismic survey hydrophone component.

5.3 Deblending Results

Based on the sail line extracted from the baseline survey, results for the iterative deblending and the deblending by compressive sensing are presented. Sections 5.3.1 and 5.3.2 show results for both methods with the pressure component (hydrophone). Section 5.3.4 shows the result for the CS deblending with the in-line horizontal component extracted from the accelerometer.

5.3.1 Iterative Deblending

As the iterative deblending relies on a coherence filter (equation 3.9), for the field data application, a combination of different filters were used to attenuate the blending noise in each iteration:

- F-K filter, filtering velocities below 1,500 m/s;
- Median filter with a 1D window of 5 samples, applied in the spatial direction;
- TFDN filter with a 2D window of 11 x 7 samples;
- Hard-thresholding with exponential decrease ($\tau_h = A \times 0.9^i$), where A is the input data maximum amplitude, and i is the current iteration.

The F-K filter, media filter and hard-thresholding were combined in the first iterations, since they are very effective in removing random noise with high amplitude. In the final iterations the median filter was replaced by the TFDN to preserve the signal and remove the remaining blending noise.

Figure 5.4 shows the iterative deblending result on one receiver gather extracted from the field data. The reconstructed[43] blended data and the intermediate results for iteration 20 and 40 are in Figure 5.4b, c, and d. From those figures, the reduction of blending noise after each iteration due to the iterative approach is clear. Analyzing the fully deblended

[43]blended data after applying the pseudo-inverse operator

data (Figure 5.4e), one can observe how effective this deblending method is when compared with the original unblended data (Figure 5.4a). The methods effectiveness becomes more evident when we evaluate the deblending error (Figure 5.4f), in which only low amplitude residual noise remains.

The F-K spectra for the unblended, blended and deblended data (after 20 iterations and final) can be seen in Figure 5.5. Comparing the unblended and deblended data F-K spectra, we can observe that the spectrum was successfully reconstructed, which confirms the result seen in the time-space domain.

Figure 5.6 shows the same steps illustrated on Figure 5.4, but the results are now shown in the shot domain. Since blending noise appears in the shot domain as coherent events, the amplitudes observed in Figure 5.6f illustrate the part of the signal that was not correctly decoded, and therefore, still contaminates the deblended data. If we compare with the residuals obtained from the synthetic data (Figure 4.12f), one can see that the application of iterative deblending on field data is more challenging. Additionally, since the effectiveness of the method relies on the parameterization of the coherence filters, this can become challenging to apply on large and complex datasets.

5.3.2 Deblending by Compressive Sensing

As demonstrated in Chapter 4, the success of deblending by compressive sensing relies on the data sparsity in some sparse domain. Before presenting the results of the method, it is crucial to investigate the field data sparsity in different domains.

Following the same methodology used with the synthetic data (Section 4.3.2), the field data was transformed into 3 different sparse domains (Fourier, Wavelet and Curvelet), and later reconstructed, considering only the k-largest coefficients. The objective of this analysis is to find the minimum number of coefficients necessary to reconstruct the signal with 99.9% of accuracy. Figure 5.7 illustrates the reconstruction analysis considering all the data (3D transforms), or only one receiver (2D transforms).

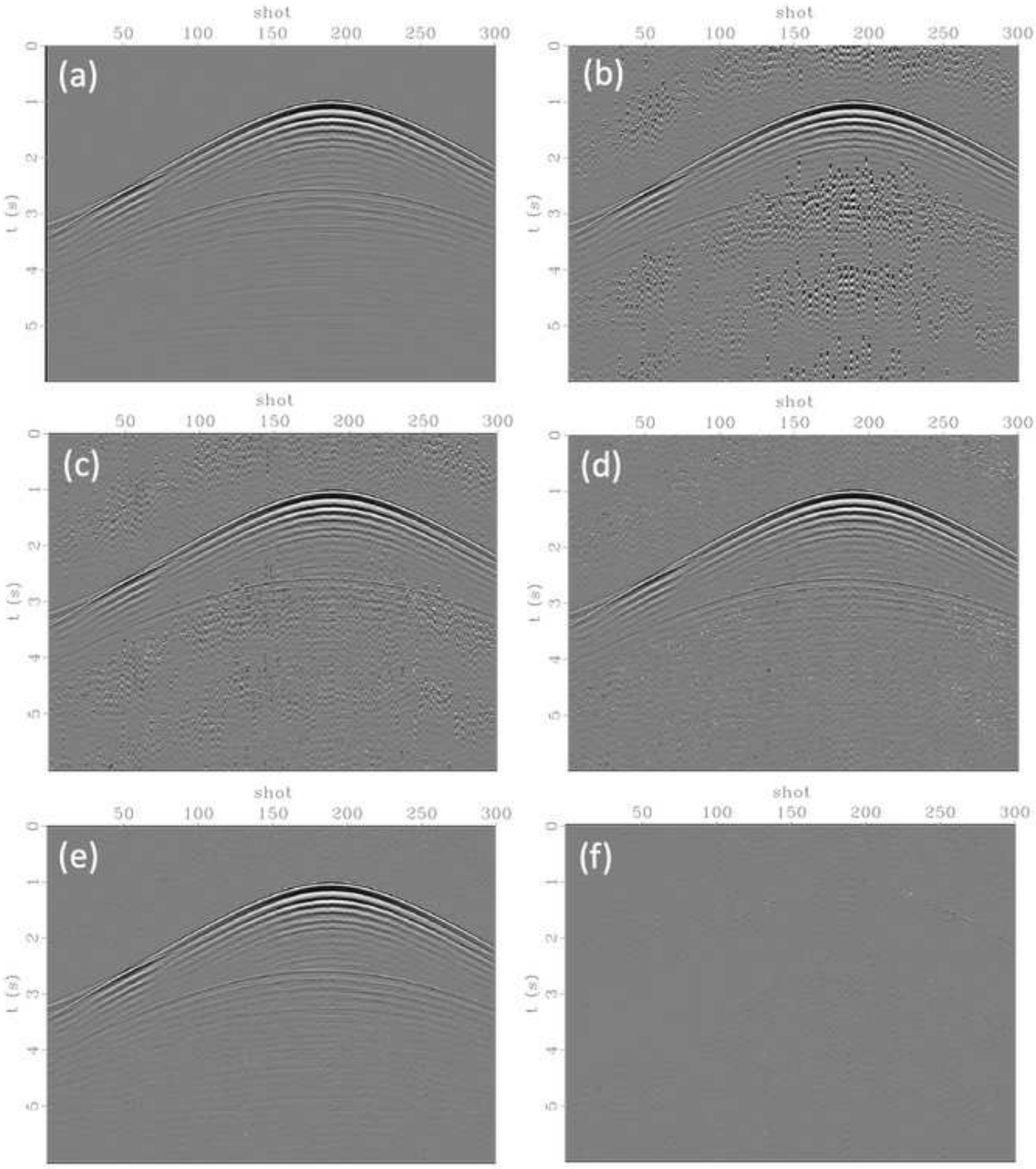

Figure 5.4: **Iterative deblending** results in the common receiver domain for the **hydrophone component**. Original unblended data (a), blended data (b), deblended data after 20 iterations (c), deblended data after 40 iterations (d), final deblended data (e), and deblending error (f). All plots have amplitude clipping to highlight the noise.

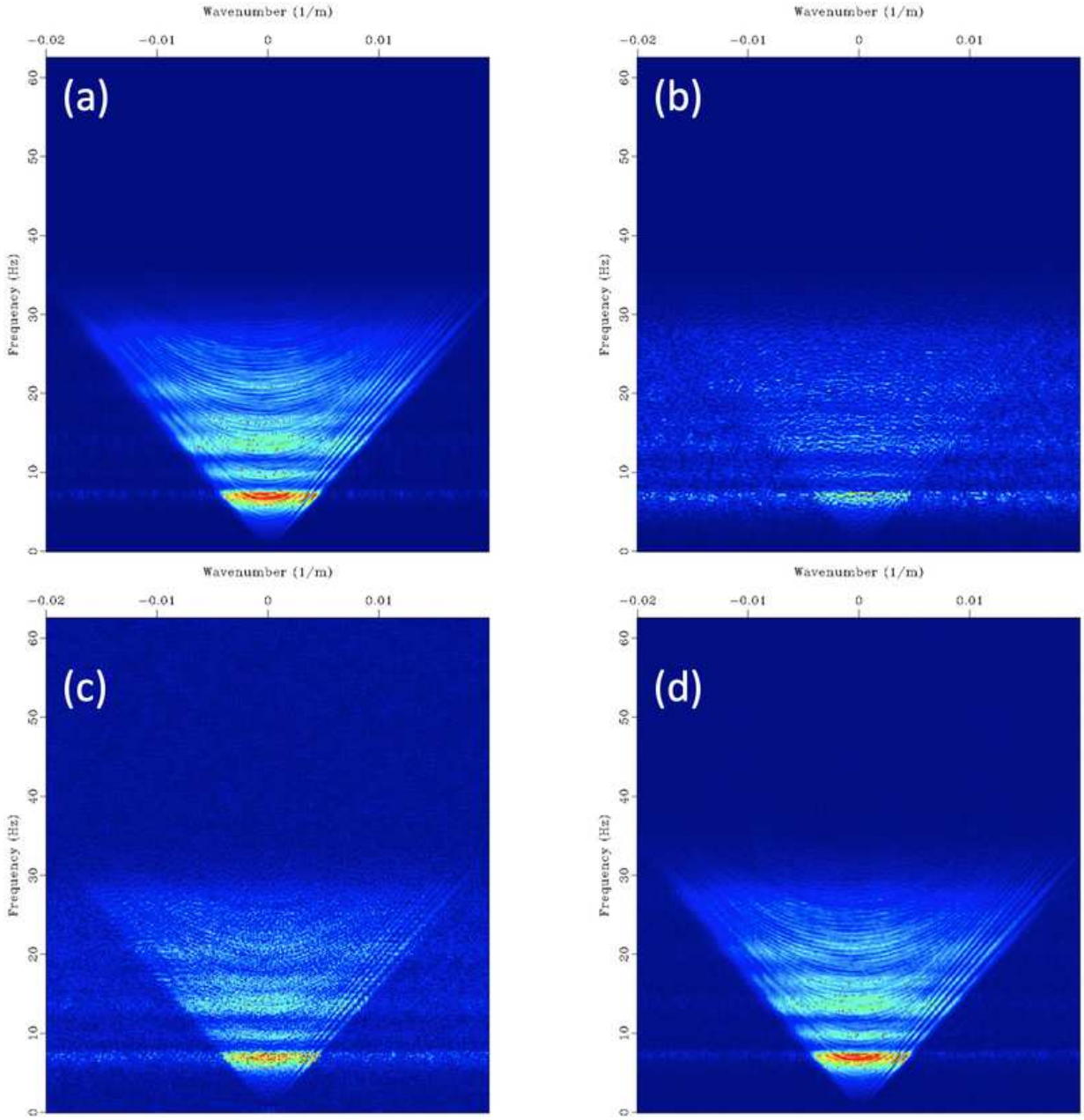

Figure 5.5: **Iterative deblending** results in the common receiver domain (F-K spectra) for the **hydrophone component**. Original unblended data (a), blended data (b), deblended data after 20 iterations (c), and final deblended data (d).

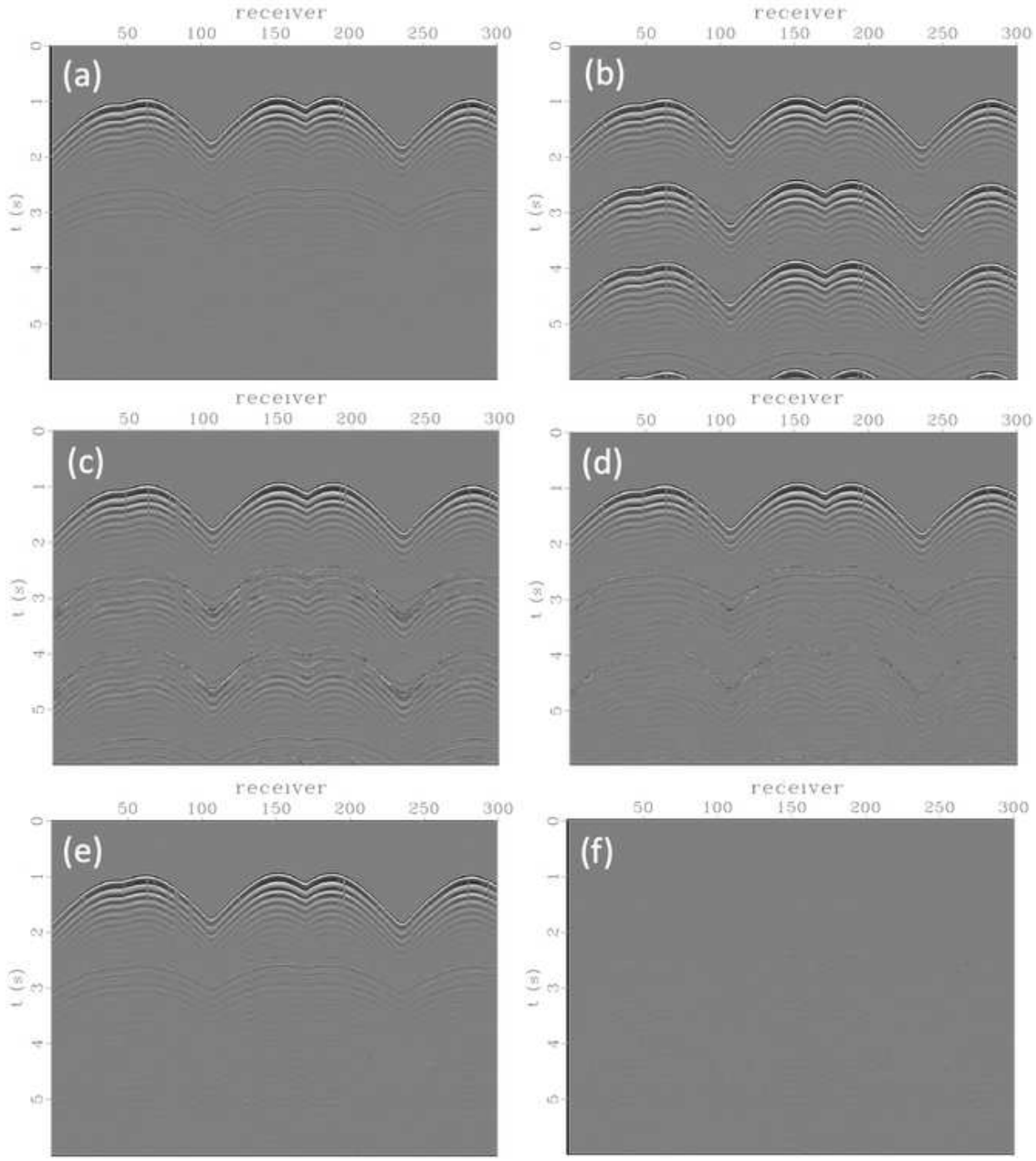

Figure 5.6: **Iterative deblending** results in the common shot domain for the **hydrophone component**. Original unblended data (a), blended data (b), deblended data after 20 iterations (c), deblended data after 40 iterations (d), final deblended data (e), and deblending error (f). All plots have amplitude clipping to highlight the noise.

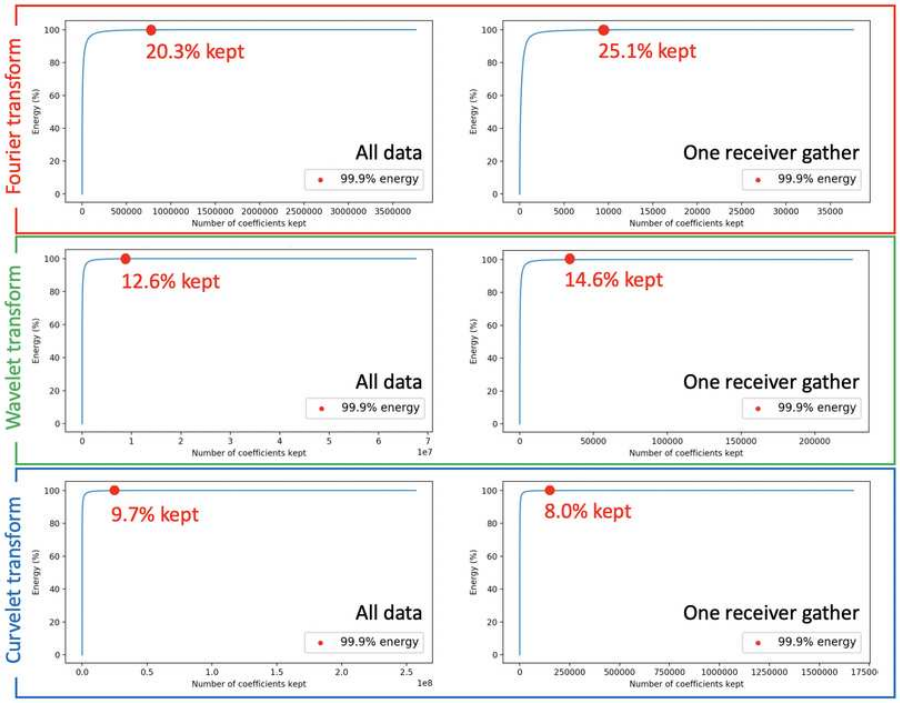

Figure 5.7: Field data sparsity analysis in Fourier, Wavelet and Curvelet domains, considering one receiver gather or the entire dataset.

Similar to the results obtained with synthetic data, one observes that the Fourier domain is the least sparse of the three transforms tested (20.3% of the coefficients are necessary to reconstruct the signal with 99.9% of accuracy). However, unlike the synthetic data results, where the sparsity of the wavelet and curvelet transform were similar, for field data, the curvelet transform is much more sparse. Using curvelets, only 9.7% of the coefficients are necessary to reconstruct the data with the target accuracy, while 12.6% are necessary with wavelets. This is an expression of how much more sparse are the data when curvelets are used as a base for data representation, which increases the probability of recovering a proper solution in this domain.

Figure 5.8 shows the deblending by compressive sensing applied to all data simultaneously in the Fourier domain to illustrate how the low sparsity of the data in this domain affects the solution of equation 3.21. The solution was achieved after 1,600 iterations of the SPGℓ_1 solver, for a misfit limit of $\sigma = 0.01$ to find the solution for equation 3.21 (repeated bellow):

$$\min_{\mathbf{f}} \|\mathbf{f}\|_1 \text{ subject to } \left\|\mathbf{b} - \mathbf{T}\mathbf{S}^H\mathbf{f}\right\|_2 \leq \sigma.$$

From the figure, one observes that the obtained solution is not adequate, with blending noise and artifacts contaminating the data. The parameters used for the test with the real data are the same used in the tests with synthetic data, including zero paddings. Additionally, changes in the misfit parameter did not affect the quality of the solution obtained. Disregarding other possible causes for the poor performance of the compressive sensing deblending in this case, we can conclude that this is directly related to the low sparsity of the data in the Fourier domain (Figure 5.7).

Repeating the test, but now looking for solutions in the Curvelet domain applied on common receiver gathers, it is possible to obtain better quality results. The solution for each common receiver gather was achieved after 1400 iterations (for each receiver gather) of the SPGℓ_1 solver, for a misfit limit of $\sigma = 0.01$. Figure 5.9 illustrates the results obtained.

The new results achieved using the curvelet transform (Figure 5.9c) are superior to those obtained with the Fourier transform (Figure 5.8c), although some residual noise is still present (Figure 5.9d). The conclusion is that the use of the appropriate sparse domain is essential for this deblending method to succeed

5.3.3 Comparative Analysis

Considering the results from the previous sections, it is essential to compare the methods. Table 5.2 shows a comparative analysis for the application for both methods on the same dataset, with the same number of CPUs, and on the same workstation. Items such as overall quality, time, difficulty in parameterization, and the need for computational resources were evaluated.

In terms of quality, both methods results are very similar, eliminating most of the blending noise. The field data results are compatible with those obtained with synthetic data, but with a higher noise level, because of its inherent complexity. Based on that, the conclusions derived from the synthetic data can be extended to the field data.

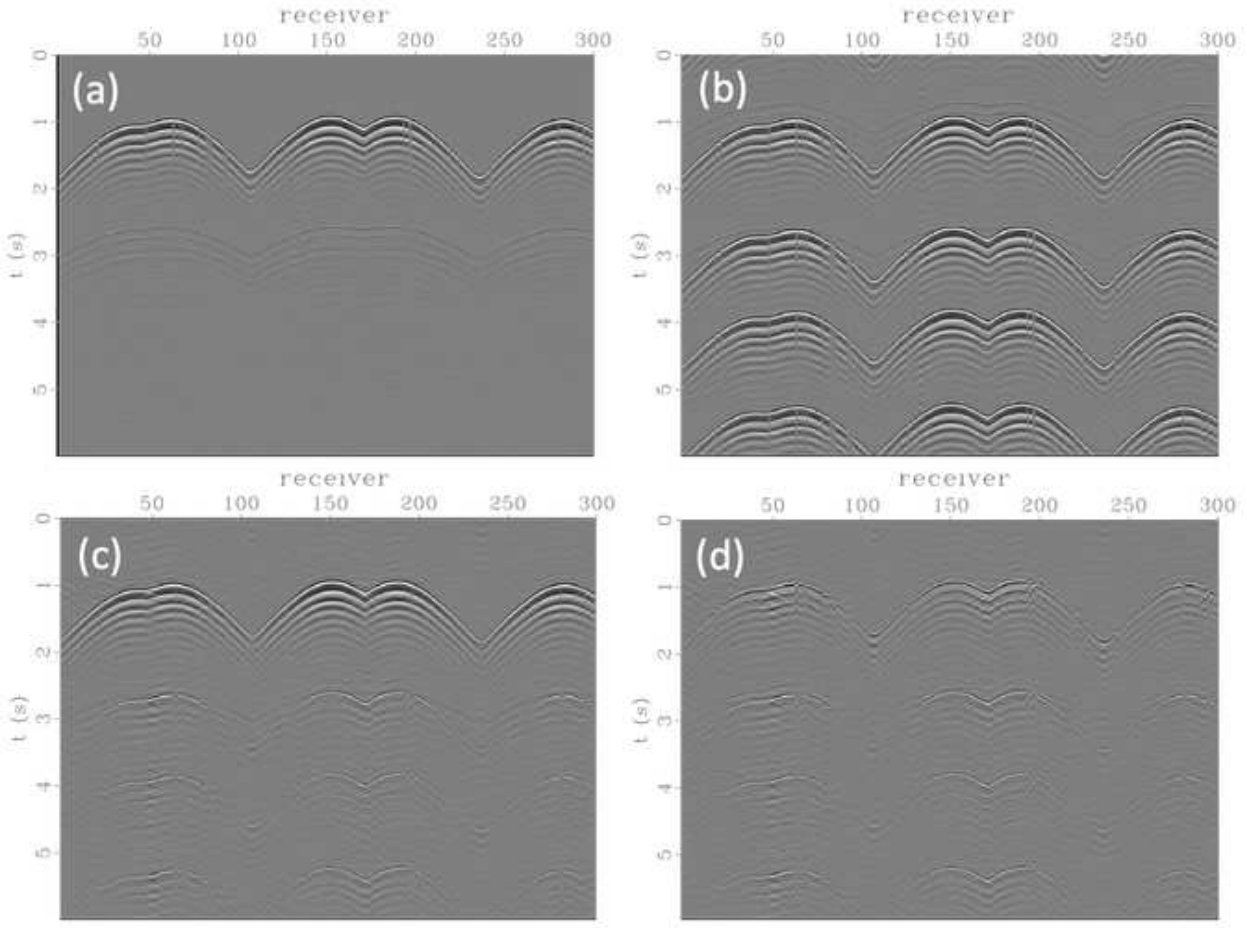

Figure 5.8: **Deblending by compressive sensing** results in the common shot domain for the **hydrophone component** using **Fourier transform**. Original unblended data (a), Blended data (b), Deblended data (c), and Deblending error (d). All plots have amplitude clipping to highlight the noise.

Table 5.2: Deblending methods comparative analysis on field data

Deblending method	**Iterative**	**CS (Fourier)**	**CS (Curvelet)**
Results quality	High, with minor residual noise	Low, with several artifacts	High, with minor residual noise
Execution time	20h	42h	300h
Memory requirement	Low (less than 2 Gb of RAM)	Medium (around 16 Gb of RAM)	High (more than 32 Gb of RAM)
Parameterization	Hard (Coherence filter parameters)	Easy (solver parameters)	Easy (solver parameters)

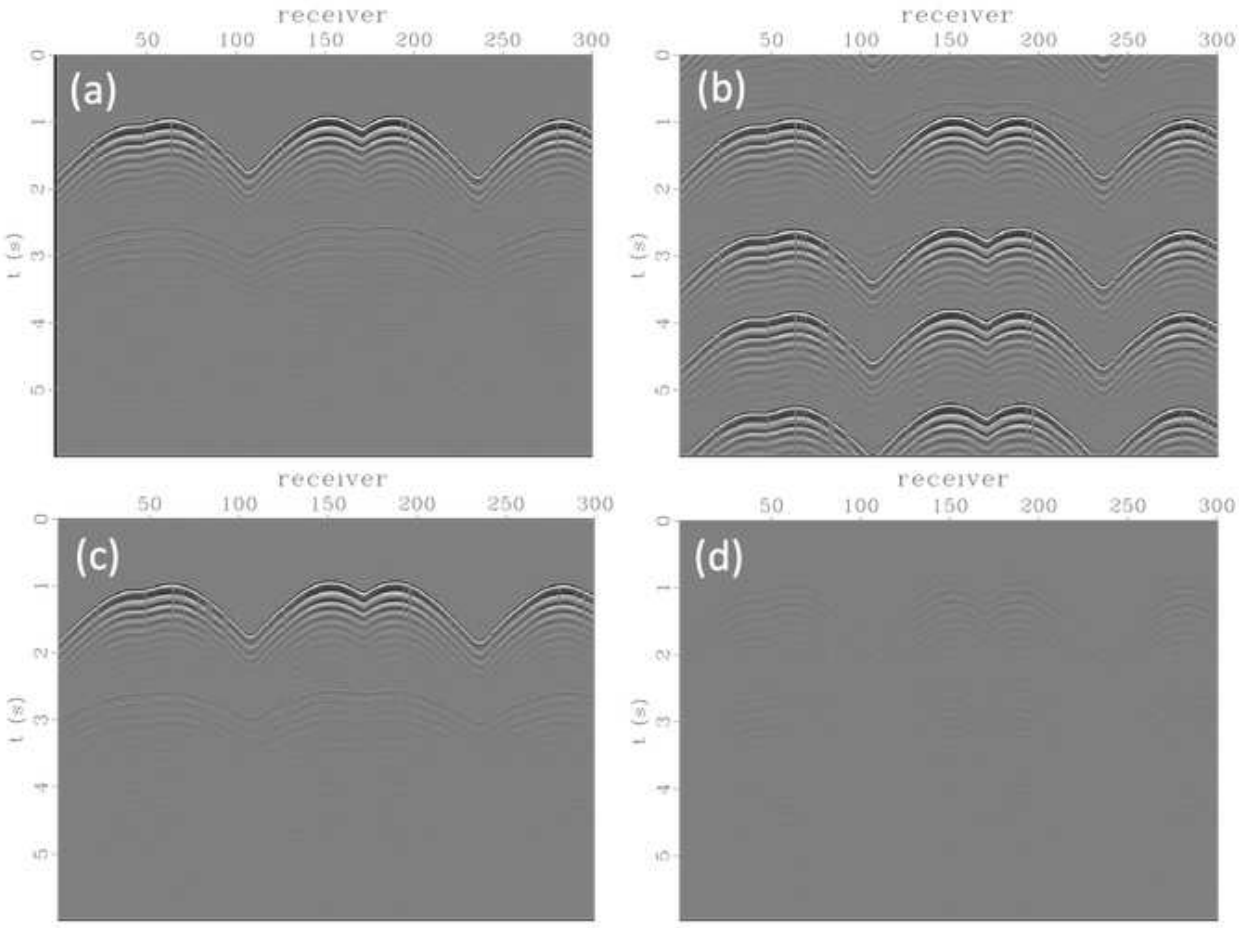

Figure 5.9: **Deblending by compressive sensing** results in the common shot domain for the **hydrophone component** using **Curvelet transform**. Original unblended data (a), Blended data (b), Deblended data (c), and Deblending error (d). All plots have amplitude clipping to highlight the noise.

In terms of run time and need for computational resources, the iterative deblending appears as the best choice since it requires less powerful computers, and it runs faster. On the other hand, the iterative method is more challenging to parametrize, since it relies on coherence filters than need to be adjusted during the iterations.

For production projects, the CS approach has the potential to generate more consistent results. However, the chosen *solver* would require some optimization in terms of execution time and memory consumption to make it compatible with the timeline for the projects.

5.3.4 Deblending by Compressive Sensing - Accelerometer Component

In the previous sections, the deblending methods were compared, considering only the hydrophone component. Based on the need to fully analyze the deblending methods on multi-component, I extended the tests to the acceleration measurements. For this, I extracted the same sail-line, the horizontal component in the in-line direction.

Considering the results of the previous sections, I used Curvelets as sparse transform applied on common receiver gathers. The solution for each common receiver gather was achieved after 1400 iterations (for each receiver gather) of the SPGℓ_1 solver, for a misfit limit of $\sigma = 0.01$. I adopted the same curvelet transform parameters used with the hydrophone. Figure 5.10 illustrates the results obtained.

The result obtained with the application of CS deblending on the acceleration component (Figure 5.10c) is equivalent to those obtained with the pressure component (Figure 5.9c). Residual noise levels are low (Figure 5.10d), and the differences between the unblended (Figure 5.10a) and deblended data (Figure 5.10c) are barely noticeable. Based on that, the conclusions derived from the synthetic data, and from the hydrophone component can be extended to the acceleration measurements. Therefore one can conclude that 4C seismic data can be deblended without major problems.

The computational resources required by the CS deblending, when applied to the accelerometer component, are equivalent to those required by the hydrophone. The execution time was about 300 hours, and the memory use was very high (more than 32 Gb of RAM).

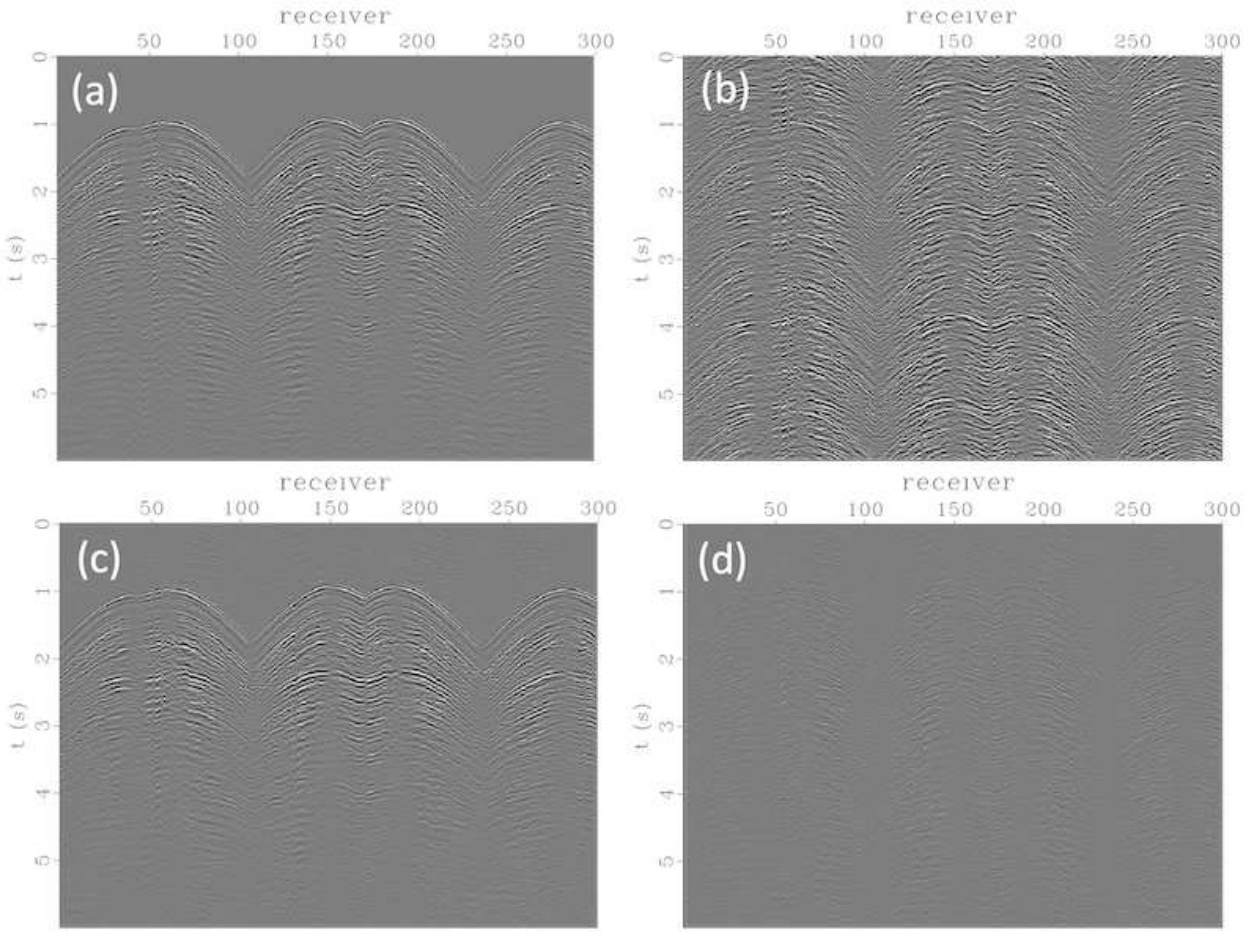

Figure 5.10: **Deblending by compressive sensing** results in the common shot domain for the **accelerometer in-line component** using **Curvelet transform**. Original unblended data (a), Blended data (b), Deblended data (c), and Deblending error (d). All plots have amplitude clipping to highlight the noise.

5.4 Field Data Analysis - Final Remarks

The application of geophysical methods in field data always brings an additional challenge to the tests because of the inherent complexity of the field data. However, the results presented in this chapter, with field data, reinforced the effectiveness of deblending techniques, similar to those obtained with synthetic data. These methods are not capable of generating a completely noise-free solution, however the remaining noise level is negligible.

In terms of quality, both methods results are very similar, eliminating most of the blending noise, with the CS deblending generating "cleaner" shot gathers than the iterative deblending. However, when the run time and need for computational resources are taken into account, the iterative deblending appears as the best choice since it requires less powerful computers, and it runs faster.

The application of the CS deblending using not only the hydrophone but also the accelerometer component brought essential insights into how effective this method is for 4C projects.

Unlike what was done with synthetic data, a 4D analysis of the deblending methods was not performed with the field data. Because, it would require to process not only a sail line but a full set of them, from the base and monitor data, for the original and deblended datasets. This would be a huge effort, since several processing steps, such as receiver reorientation, PZ summation, 4D binning, migration, and amplitude matching would be necessary for a fair 4D analysis to be performed. Nevertheless, based on the conclusion with synthetic data, the impact of blending noise on real 4D projects is expected to be very low.

Based on the obtained NRMS level, I consider 0.5% to be the minimum achievable NRMS value for our 4D blended project. If we extrapolate the modeling results to the background NRMS extracted from the Jubarte seismic data (which was 4.5% above the reservoir), we can conclude that opting to repeat the Jubarte seismic survey as a blended acquisition would increase the NRMS values for a time-lapse project to only 5%.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

In this chapter, I talk about my insights and conclusions about seismic deblending. I also present some recommendations about future work that can be developed to further the understanding of deblending methods for time-lapse projects.

6.1 Deblending on synthetic data

The first results obtained in this project, and presented in Chapter 4, were obtained through the application of deblending techniques using synthetic data. To make the initial tests more realistic, the synthetic data were modeled considering the properties and geological features from a real production field, the Jubarte Field. The modeling also incorporated realistic scenarios in terms of acquisition geometries. An OBS geometry, with a source vessel detonating seismic sources in shorter time intervals than usual, simulated a situation of increased productivity, which is often desired by companies.

Analyses of the results demonstrated the effectiveness of both deblending techniques, iterative deblending and deblending by compressive sensing, for successfully recovering deblended data, although minor artifacts remained. Similar to most seismic processing techniques, these methods are not capable of generating an entirely noise-free solution, even when using synthetic data, although other implementations may produce different results (for better or worse). In practice, the remaining noise/artifacts produce marginal effects in the final image, as was demonstrated.

A comparison of deblending methods also demonstrated how a priori information plays an essential role in inverse problems. In this case, the prior knowledge that the solutions are sparse is vital for the success of using either method.

In terms of quality, both methods are very similar in their ability to eliminate most of the blending noise, with the CS deblending generating slightly better results than the iterative

deblending. The RMS plots presented in Section 4.3.3 corroborate this conclusion.

Considering that the quality of the results must be analyzed along with other aspects of the deblending process, such as the amount of computational resources and the time required for the application of the methods, the iterative deblending appears the best choice, since it requires less powerful computers and runs faster. For projects with higher complexity, the compressive sensing approach has the potential to generate more consistent results, however its application would require optimization in terms of running time and memory consumption.

Going further in the analysis, the results obtained after the introduction of random noise before the seismic deblending, brought to light additional relevant insights and made the simulation more realistic. The deblending method tested in this scenario generated results with similar quality to those obtained without noise; that is, the deblending process was able to attenuate blending noise and random noise simultaneously. Variations in noise levels and distribution could affect these conclusions since, in practice, they can change from one shot to the next. This result is not a surprise since the foundation for most deblending methods is the sparsity of the data in sparse domains. Sparse domains are frequently used in denoising processes, where the elimination of less relevant coefficients, generally associated with noise, is fundamental to any sparse recovery method. Variations in noise intensity and distribution were not simulated. Therefore the conclusions can be slightly different if more complex noise patterns are analyzed (for example, noise level changing from one shot to the other).

6.2 Deblending on field data

The results presented in Chapter 5, with field data, reinforced the effectiveness of deblending techniques for successfully recovering deblended data. As before, minor artifacts remained, since the methods were not capable of generating an entirely noise-free solution; however, the remaining noise level is negligible.

The test with field data reinforced the importance of properly select a sparse transform to recover deblended data with good quality. The analysis with different sparse transforms, such

as Fourier, Wavelet, and Curvelet transforms, demonstrated that the Fourier and Wavelet transforms are not sparse enough for seismic deblending applications with field data. The results, when Fourier and Wavelet transforms were used, were highly contaminated with blending noise, while with the Curvelet transform, only minor artifacts were present.

Testing the horizontal component of the accelerometer, and not just the pressure component, brought essential insights into understanding how blended acquisitions affect the directional components. Different from the hydrophone component tested previously, the horizontal component of the accelerometer contained significant lateral amplitude variations and less coherent events. Nevertheless, the tested deblending method delivered results with good quality and minor artifacts.

Based on the results, it is possible to conclude that 4C data can be deblended without major difficulties, as soon as the necessary computational resources are available.

From my analysis, using a 32 CPUs workstation with 128 Gb of RAM, it would be required 16 days to deblend a 4C sail-line (running jobs in parallel) as the one tested. Since the projects' timeline is in general very tight, the application of seismic deblending on 4C data would require optimization in terms of running time and memory consumption.

6.3 Time-lapse analysis (on synthetic data)

Considering the objectives of the project, the 4D analysis of synthetic data provided valuable insight into how blending noise and deblending techniques may influence 4D projects. The repeatability analysis that was carefully conducted, isolating external effects such as uncertainty in velocity models or differences in migration algorithms, was one of the most relevant parts of the work.

Moreover, the repeatability measurement using the NRMS amplitude, demonstrated for our data that the impact on the 4D signal is negligible; for a 20-25% NRMS change expected from production, the impact on the 4-D signal is of the order 0.5%. Similarly, the background NRMS amplitude that was approximately 3.3% was increased by 0.5% to 3.8% after deblending. If we extrapolate the modeling results to the background NRMS extracted from

the Jubarte seismic data (which was 4.5% above the reservoir), we can conclude that opting to repeat the Jubarte seismic survey as a blended acquisition would increase the NRMS values for a time-lapse project to only 5%.

6.4 Recommendations

Although two seismic deblending methods have been extensively tested in the development of the project, several other methods can be explored and compared. Future work should include the generalization of the iterative deblending process using shaping regularization (Chen *et al.*, 2014b), or the matrix completion formulation, which takes advantage of low-rank matrices (Kumar *et al.*, 2015a). An in-depth analysis of different methods is essential for a more comprehensive understanding of the impact of seismic deblending on 4D projects.

During the project development, I observed that the effectiveness of the deblending methods directly correlates to the sparsity of the data. Although the tests indicated that the deblending techniques are independent of the characteristics of the medium, the studies carried out with synthetic data considered only an isotropic medium with little geological complexity. Even tests with field data were based on data collected from a weakly anisotropic medium. Therefore, future work should analyze the effectiveness of these methods on data modeled on or acquired from highly anisotropic and heterogeneous media.

As the field data available for the project were acquired using a traditional geometry, and from that blended numerically, the analyzes were not able to assess the effectiveness of the methods with a truly blended acquisition. Thus, future work should use real blended field data, since challenges associated with firing time recording precision, increased vessel speed, and field obstructions are expected to affect the deblended data final quality.

The 4D analysis carried out in this thesis considered only synthetic data because to repeat the study with field data would require the processing of not only one sail-line but several sail-lines from the base and monitor data. This processing would entail a considerable effort since several processing steps, such as reorienting the receivers, PZ summation, 4D

binning, migration, and amplitude matching would be necessary for a fair 4D analysis to be performed. For a more conclusive analysis of the effects of seismic deblending on 4D projects, field data that has gone through the deblending process must be processed and analyzed in the migrated image domain. To make this effort compatible with the resources available in academia, an analysis using 2D seismic data could be executed

Finally, if we consider that the main methods of quantitative interpretation are based on amplitude versus offset analysis (AVO), the effects of residual blending noise need to be analyzed quantitatively on pre-stack migrated data.

REFERENCES CITED

Aaron, Peter, Byerley, Grant, & Monk, David. 2016. Assessing marine 3D seismic acquisition with new technology: A case history from Suriname. *Pages 41–45 of: SEG Technical Program Expanded Abstracts 2016*. Society of Exploration Geophysicists.

Abadir, Karim M, & Magnus, Jan R. 2005. *Matrix algebra*. Vol. 1. Cambridge University Press.

Antonini, Marc, Barlaud, Michel, Mathieu, Pierre, & Daubechies, Ingrid. 1992. Image coding using wavelet transform. *IEEE Transactions on image processing*, **1**(2), 205–220.

Barr, Fred J. 1997. Dual-sensor OBC technology. *The Leading Edge*, **16**(1), 45–52.

Barros, Caio. 2015. *Internal Processing Report - 4D/4C Jubarte PRM*. unpublished.

Beasley, Craig J., Chambers, Ronald E., & Jiang, Zerong. 1998. A new look at simultaneous sources. *1998 SEG Annual Meeting*.

Becker, Stephen, Bobin, Jérôme, & Candès, Emmanuel J. 2011. NESTA: A fast and accurate first-order method for sparse recovery. *SIAM Journal on Imaging Sciences*, **4**(1), 1–39.

Berkhout, A J. 1982. *Imaging of Acoustic Energy by Wave Field Extrapolation, A. Theoretical Aspects*. Elsevier.

Berkhout, A J. 2008. Changing the mindset in seismic data acquisition. *The Leading Edge, 27*, 924–938.

Birgin, Ernesto G, Martínez, José Mario, & Raydan, Marcos. 2000. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, **10**(4), 1196–1211.

Blacquière, Gerrit, Berkhout, Guus, & Verschuur, Eric. 2009. Survey design for blended acquisition. *Pages 56–60 of: SEG Technical Program Expanded Abstracts 2009*. Society of Exploration Geophysicists.

Caldwell, Jack, & Dragoset, William. 2000. A brief overview of seismic air-gun arrays. *The leading edge*, **19**(8), 898–902.

Caldwell, Jack, Koudelka, Eric, Nesteroff, Kenneth, Price, Russel, & Zhang, Peter. 2015. Seismic permanent reservoir monitoring (PRM)–A growing market. *First Break*, **33**(11), 65–73.

Candes, Emmanuel J, & Donoho, David L. 2000. *Curvelets: A surprisingly effective non-adaptive representation for objects with edges*. Tech. rept. Stanford Univ Ca Dept of Statistics.

Candes, Emmanuel J, Romberg, Justin K, & Tao, Terence. 2006. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, **59**(8), 1207–1223.

Candes, Emmanuel J, *et al.* . 2008. The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathematique*, **346**(9-10), 589–592.

Chen, Hongling, Cao, Siyuan, Zu, Shaohuan, Yang, Bo, Shen, Shian, & Sun, Xiaoming. 2018. Iterative deblending using the POCS algorithm in the approximate flattened domain. *Journal of Geophysics and Engineering*, **15**(4), 1104–1121.

Chen, Yangkang, Fomel, Sergey, & Hu, Jingwei. 2013. Iterative deblending of simultaneous-source seismic data using seislet-domain shaping regularization. *Geophysics*, **79**(5), V179–V189.

Chen, Yangkang, Yuan, Jiang, Jin, Zhaoyu, Chen, Keling, & Zhang, Lele. 2014a. Deblending using normal moveout and median filtering in common-midpoint gathers. *Journal of Geophysics and Engineering*, **11**(4), 045012.

Chen, Yangkang, Fomel, Sergey, & Hu, Jingwei. 2014b. Iterative deblending of simultaneous-source seismic data using seislet-domain shaping regularization. *Geophysics*, **79**(5), V179–V189.

Chen, Yangkang, Yuan, Jiang, Zu, Shaohuan, Qu, Shan, & Gan, Shuwei. 2015. Seismic imaging of simultaneous-source data using constrained least-squares reverse time migration. *Journal of Applied Geophysics*, **114**, 32–35.

Cheng, Jinkun, & Sacchi, Mauricio D. 2015. Separation and reconstruction of simultaneous source data via iterative rank reduction. *Geophysics*, **80**(4), V57–V66.

Damasceno, Andrea. 2020. *4D Quantitative Interpretation of Jubarte Field (Brazil) - an Integrated Approach*. Ph.D. thesis, Colorado School of Mines. Arthur Lakes Library.

Dariva, Paula, Ramos Filho, Wilson, Born, Christian, Falcone, Célia, & Zorzanelli, Isabella. 2016 (10). Monitoramento Sísmico 4D no Campo de Jubarte, Bacia de Campos - Brasil. *In: Rio Oil and Gas 2016 Expo and Conference.*

Donoho, David L. 2006. Compressed sensing. *IEEE Transactions on information theory*, **52**(4), 1289–1306.

Duey, R. 1996. Triple-vessel system boosts productivity in the North Sea: Hart's Show. *In: Special Edition for the 66th Annual International Meeting, SEG.*

Duey, Rhonda. 2011. 'Flying'nodes shift marine seismic paradigm. *Hart's E & P*, **84**(12).

Elboth, Thomas, Geoteam, Fugro, Qaisrani, Hamid Hayat, Hertweck, Thomas, & Imaging, Fugro Seismic. 2008. De-noising seismic data in the time-frequency domain. *SEG Las Vegas 2008 Annual Meeting*, 2622–2626.

Fomel, Sergey, Sava, Paul, Vlad, Ioan, Liu, Yang, & Bashkardin, Vladimir. 2013. Madagascar: Open-source software project for multidimensional data analysis and reproducible computational experiments. *Journal of Open Research Software*, **1**(1).

Gan, Shuwei, Wang, Shoudong, Chen, Yangkang, Chen, Xiaohong, & Xiang, Kui. 2016. Separation of simultaneous sources using a structural-oriented median filter in the flattened dimension. *Computers & Geosciences*, **86**, 46–54.

Hennenfent, Gilles, & Herrmann, Felix J. 2006. Seismic denoising with nonuniformly sampled curvelets. *Computing in Science & Engineering*, **8**(3), 16–25.

Huang, Weilin, Wang, Runqiu, Gong, Xiangbo, & Chen, Yangkang. 2018. Iterative deblending of simultaneous-source seismic data with structuring median constraint. *IEEE Geoscience and Remote Sensing Letters*, **15**(1), 58–62.

Kragh, ED, & Christie, Phil. 2002. Seismic repeatability, normalized rms, and predictability. *The Leading Edge*, **21**(7), 640–647.

Kumar, Rajiv, Da Silva, Curt, Akalin, Okan, Aravkin, Aleksandr Y., Mansour, Hassan, Recht, Benjamin, & Herrmann, Felix J. 2015a. Efficient matrix completion for seismic data reconstruction. *Geophysics*, **80**(5), V97–V114.

Kumar, Rajiv, Wason, Haneet, & Herrmann, Felix J. 2015b. Source separation for simultaneous towed-streamer marine acquisition — A compressed sensing approach. *Geophysics*, **80**(6), WD73–WD88.

Laws, RM, Kragh, E, & Morgan, G. 2008. Are seismic sources too loud? *In: 70th EAGE Conference and Exhibition incorporating SPE EUROPEC 2008.*

Le Montagner, Yoann, Angelini, Elsa, & Olivo-Marin, Jean-Christophe. 2011. Comparison of reconstruction algorithms in compressed sensing applied to biological imaging. *Pages 105–108 of: 2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*. IEEE.

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

Lin, Tim TY, & Herrmann, FJ. 2009. Designing simultaneous acquisitions with compressive sensing. *Pages cp–127 of: 71st EAGE Conference and Exhibition incorporating SPE EUROPEC 2009*. European Association of Geoscientists & Engineers.

Luenberger, David G. 1997. *Optimization by vector space methods*. John Wiley & Sons.

Mahdad, Ahraz. 2012. *Deblending of seismic data*. Uitgeverij BOXPress, Oisterwijk, The Netherlands.

Mahdad, Araz, Doulgeris, Panagiotis, & Blacquiere, Gerrit. 2011. Separation of blended data by iterative estimation and subtraction of blending interference noise. *Geophysics*, **76**(3), Q9–Q17.

Olofsson, Björn, Massacand, Christophe, & Millington, Mike. 2007. Determining 3-C geophone orientation from a single seismic shot. *Pages 970–974 of: SEG Technical Program Expanded Abstracts 2007*. Society of Exploration Geophysicists.

Peng, Can, & Meng, Jie. 2016. Inversion-based 3D deblending of towed-streamer simultaneous source data using sparse Taup and wavelet transforms. *Pages 4607–4611 of: SEG International Exposition and 86th Annual Meeting*.

Penrose, R. 1955. A generalized inverse for matrices. *Mathematical Proceedings of the Cambridge Philosophical Society*, **51**(3), 406–413.

Thedy, EA, Filho, Ramos, Johann, WL, Seth, SN, Souza, SC, & Murray, PE. 2013. Jubarte Permanent Reservoir Monitoring–Installation and First Results. *Pages 918–920 of: 13th International Congress of the Brazilian Geophysical Society & EXPOGEF, Rio de Janeiro, Brazil, 26–29 August 2013*. Society of Exploration Geophysicists and Brazilian Geophysical Society.

Threadgold, Ian M, Zembeck-England, Kristin, Aas, Per Gunnar, Fontana, Philip M, Hite, Damian, & Boone, William E. 2006. Implementing a wide azimuth towed streamer field trial: the what, why and mostly how of WATS in Southern Green Canyon. *Pages 2901–2904 of: SEG Technical Program Expanded Abstracts 2006*. Society of Exploration Geophysicists.

van den Berg, E., & Friedlander, M. P. 2019 (December). *SPGL1: A solver for large-scale sparse reconstruction*. https://friedlander.io/spgl1.

Van Ewout Berg, D. E.N., & Friedlander, Michael P. 2008. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, **31**(2), 890–912.

Vermeer, Gijs J O. 2002. 3-D Seismic Survey Design. *Society of Exploration Geophysicists, Tulsa*.

Wason, Haneet. 2017. *Simultaneous-source seismic data acquisition and processing with compressive sensing*. Ph.D. thesis, University of British Columbia.

Yang, Junfeng, & Zhang, Yin. 2011. Alternating direction algorithms for \ell_1-problems in compressive sensing. *SIAM journal on scientific computing*, **33**(1), 250–278.

APPENDIX

ALGORITHMS

Listing A.1: Iterative deblending algorithm

```
#!/usr/bin/env python
import m8r as sf
import numpy as np
import os
import time
import scipy.sparse as sparse
import scipy.sparse.linalg as linalg
import scipy.signal as signal
import sys

def fft1D(data_t,L):
    data_fft = np.fft.fft(data_t,L)
    data_fft = data_fft[:,0:L//2+1]
    data_fft[:,1:] = 2.0/L*data_fft[:,1:]
    return np.abs(data_fft),np.angle(data_fft)

def ifft1D(amp,phase,L,tsamples):
    data_fft = amp*np.exp(1j*phase)
    data_fft[:,1:] = L/2.0*data_fft[:,1:]
    data_fft = np.hstack((data_fft,np.fliplr(data_fft[:,1:])))
    data_fft[:,L//2+1:] = np.real(data_fft[:,L//2+1:])-1j*np.imag(
   data_fft[:,L//2+1:])
    data_t = np.real(np.fft.ifft(data_fft,L))
    return data_t[:,0:tsamples]

def fft2D(input,L):
    L2 = L[1]//2+1
    output = np.fft.fftn(input,s=L,norm='ortho')
    output = np.fft.fftshift(output,axes=(0,)) #Center the wavenumbers
    output = output[:,:L2]
    return output

def ifft2D(input,L,n_shots,n_times):
    input = np.fft.ifftshift(input,axes=(0,))
    L2 = L[1]//2+1
    shift = (L[1]+1)%2 # 1 if L[1] is even
    F = np.concatenate((input, np.conjugate(np.flip(input[:,1:L2-shift
   ],axis=1))) ,axis=1)
```

```
    F[1:,L2+1:] = np.flip(F[1:,L2+shift:],axis=0)
    output = np.fft.ifftn(F,s=L,norm='ortho')
    return np.real(output[:n_shots,:n_times])

def fkfilter(data,vel,n_shots,n_times,t_sample_rate,x_sample_rate,L,
    hi_cut):
    L2 = L[1]//2+1
    output = fft2D(data,L)
    freq_t = np.fft.fftfreq(L[1],t_sample_rate)
    freq_t = freq_t[:L2]
    freq_x = np.fft.fftfreq(L[0],x_sample_rate)
    freq_x = np.fft.fftshift(freq_x)
    freq_xx, freq_tt = np.meshgrid(freq_x, freq_t)
    freq_xx = freq_xx.T
    freq_tt = freq_tt.T
    max_freq_t = np.max(freq_t)
    max_freq_x = np.max(freq_x)
    output_filtered = np.where(np.abs(freq_tt) > hi_cut,    0.0     ,
    output)
    output_filtered = np.where(freq_tt/vel <= np.abs(freq_xx),    0.0
    ,output_filtered)
    output_filtered = ifft2D(output_filtered,L,n_shots,n_times)
    return output_filtered

def mediandilter1D(data,size):
    output = signal.medfilt2d(data,(size,1))
    return output

def thresholding(data,ref_data,perc,type='hard'):
    max_amp = np.absolute(ref_data).max()
    T = perc*max_amp
    if type=='hard':
        output = np.where(np.absolute(data) > T, data, 0.0 )
    else:
        output = np.where(np.absolute(data) > T, data-T, np.where(np.
    absolute(data) < -T, data+T, 0.0))
    return output

def tfdnfilter(rec_gather_t_in,n_shots,tsamples,w1,w2,threshold = 1.0):
    rec_gather_t_out = np.zeros_like(rec_gather_t_in)
    L = 2*w1+1
    sys.stderr.write("\n")
    for j in np.arange(n_shots):
        sys.stderr.write("      TFDN: trace #%03d\r" % (j))
        startj = j-w2//2
        endj   = j+w2//2+1
        current_j_window = w2//2
```

```
            if (j-w2//2) < 0:
                startj = 0
                endj   = w2
                current_j_window = j
            if (j+w2//2) > (n_shots-1):
                startj = n_shots-w2
                endj   = n_shots
                current_j_window = (w2-1)-((n_shots-1)-j)

            for k in np.arange(tsamples):
                startk = k-w1//2
                endk   = k+w1//2+1
                current_k_window = w1//2
                if (k-w1//2) < 0:
                    startk = 0
                    endk   = w1
                    current_k_window = k
                if (k+w1//2) > (tsamples-1):
                    startk = tsamples-w1
                    endk   = tsamples
                    current_k_window = (w1-1)-((tsamples-1)-k)

                windowt_int = rec_gather_t_in[startj:endj,startk:endk]
                amp_in,phase = fft1D(windowt_int,L)
                amp_out = amp_in
                for l in np.arange(w1):
                    median = np.sort(amp_in[:,l])[w2//2]
                    if amp_in[current_j_window,l] > threshold*median:
                        amp_out[current_j_window,l] = threshold*median
                windowt_out = ifft1D(amp_out,phase,L,tsamples)
                rec_gather_t_out[j,k] = windowt_out[current_j_window,
    current_k_window]
        sys.stderr.write("\n")
        return rec_gather_t_out

def BuildT(n_recs,n_shots,n_times,n_sources,dither,t_sample_rate,
    time_max):
        np.random.seed(9389483)
        timeshifts =  (np.random.rand(n_shots)*dither - dither/2)
        timeshifts[0] = 0.0
        start_time = np.linspace( 0,  (n_shots-1)*time_max/n_sources,
    n_shots)
        start_time += timeshifts
        start_time = np.around(start_time/t_sample_rate).astype(int)*
    t_sample_rate
        t_blended_shot = start_time[n_shots-1]+time_max
```

```
    n_blended_times = np.int(t_blended_shot*1000)//np.int(t_sample_rate
    *1000) + 1
    elems = np.ones((n_shots*n_times))
    rows  = np.zeros_like(elems)
    cols  = np.zeros_like(elems)
    inc_shot = n_times
    for i in np.arange(n_shots):
        index_start = i*inc_shot
        index_end   = index_start+inc_shot
        row_start = int(start_time[i]/t_sample_rate)
        row_end   = row_start + inc_shot
        col_start = index_start
        col_end   = index_end
        rows[index_start:index_end] = np.arange(row_start,  row_end)
        cols[index_start:index_end] = np.arange(col_start,  col_end)
    A_temp = sparse.csr_matrix( (elems,(rows,cols)), shape=(
    n_blended_times , n_shots*n_times) )

    def Amatvec(x_vector):
        return A_temp.dot(x_vector)

    def Armatvec(b_vector):
        return (A_temp.getH()).dot(b_vector)

    A = linalg.LinearOperator((n_blended_times , n_shots*n_times),
    matvec=Amatvec, rmatvec=Armatvec)
    return A,n_blended_times

def filter(data_orig,ref_data,vel,median_size,threshold,n_shots,n_times
    ,t_sample_rate,
                           x_sample_rate, L,hi_cut,tfdn_w1,tfdn_w2,
    last_iter = False):
    data = np.copy(data_orig)
    fk_filtered = fkfilter(data,vel,n_shots,n_times,t_sample_rate,
    x_sample_rate, L,hi_cut)
    fk_filtered  = thresholding(fk_filtered,     ref_data,threshold,type
    ='hard')
    if last_iter:
        tfdn_filtered = tfdnfilter(data,n_shots,n_times,tfdn_w1,tfdn_w2
    )
        ref_filtered = thresholding(tfdn_filtered,     ref_data,
    threshold,type='hard')
    else:
        median_filtered = mediandilter1D(data,median_size)
        ref_filtered = thresholding(median_filtered,  ref_data,
    threshold,type='hard')
```

```
    return np.where(np.absolute(ref_filtered) > 0.0, fk_filtered,
    ref_filtered)
############################ MAIN ################################
RSF_IN  = sf.Input()
RSF_OUT_BLD = sf.Output()
par = sf.Par()

n_iters     = int(par.string("n_iters"))
n_sources   = int(par.string("n_sources"))
dither      = float(par.string("dither"))
median_size = int(par.string("median_size",'5'))
vel         = float(par.string("vel",'1500.0'))
threshold   = float(par.string("threshold",'0.9'))
tfdn_w1     = int(par.string("tfdn_w1",'11'))
tfdn_w2     = int(par.string("tfdn_w2",'7'))
hi_cut      = float(par.string("hi_cut",'30'))

# Read axes
a1 = RSF_IN.axis(1); n_times = a1['n']; o1  = a1['o']; t_sample_rate =
    a1['d']
a2 = RSF_IN.axis(2); n_shots = a2['n']; o2  = a2['o']; d2 = a2['d']
a3 = RSF_IN.axis(3); n_recs  = a3['n']; o3  = a3['o']; d3 = a3['d']
n_times_orig = RSF_IN.int("ntimes_orig")
n_shots_orig = RSF_IN.int("nshots_orig")
time_max = (n_times_orig-1)*t_sample_rate

#fft limits
padding = 100
n2_fft = n_shots_orig+padding
n1_fft = (n_times_orig+padding)//2+1
L = [n2_fft,n_times_orig+padding]

# Bulding matrix T
sys.stderr.write("Building matrix T\n")
T,n_blended_times = BuildT(n_recs,n_shots_orig,n_times_orig,n_sources,
    dither,t_sample_rate,time_max)
sys.stderr.write("    T Shape = (%d,%d)\n" %(A.shape[0],A.shape[1]))
sys.stderr.write("Done\n")

# Setting output dimensions
a2['n'] = n_shots_orig
a1['n'] = n_times_orig
RSF_OUT_BLD.putaxis(a1,1)
RSF_OUT_BLD.putaxis(a2,2)
RSF_OUT_BLD.putaxis(a3,3)

sys.stderr.write("Deblending data (Iteratively)\n")
```

```
for j in np.arange(n_recs):
    sys.stderr.write("Rec #%03d\n" % ( o3 + j*d3 ))
    input_data = RSF_IN.read(shape=(n_shots,n_blended_times))
    b_vector = input_data.reshape((n_blended_times,))
    THb = T.rmatvec(b_vector)  #pseudo-deblending
    THb = THb.reshape((n_shots_orig,n_times_orig))
    xii = np.copy(THb)
    last_iter = False
    for i in np.arange(1,n_iters+1):
        sys.stderr.write("    Iteration #%02d\r" % (i))
        if i == n_iters:
            last_iter = True
        Fxi_pred = filter(xii,THb,vel,median_size,threshold**i,
   n_shots_orig,n_times_orig,t_sample_rate,d2, L,hi_cut,tfdn_w1,tfdn_w2
   ,last_iter)
        Fxi_pred_vec = Fxi_pred.reshape((n_shots_orig*n_times_orig,))
        Fxi_dbl_vec = T.rmatvec(T.matvec(Fxi_pred_vec))
        Fxi_dbl = Fxi_dbl_vec.reshape((n_shots_orig,n_times_orig))
        Fxi_noise = Fxi_dbl-Fxi_pred
        xii = THb - Fxi_noise
    sys.stderr.write("\n")
    RSF_OUT_BLD.write(xii)
RSF_OUT_BLD.close()
RSF_IN.close()
sys.exit(0)
```

Listing A.2: Compressive sensing deblending algorithm

```
#!/usr/bin/env python
import m8r as sf
import numpy as np
import os
import time
import scipy.sparse as sparse
import scipy.sparse.linalg as linalg
import sys
import spgl1

def fft3D(input,L):
    L3 = L[2]//2+1
    output = np.fft.fftn(input,s=L,norm='ortho')
    output = output[:,:,:L3]
    return output

def ifft3D(input,L):
    L3 = L[2]//2+1
    shift = (L[2]+1)%2 # 1 if L[2] is even
    F = np.concatenate((input, np.conjugate(np.flip(input[:,:,1:L3-
    shift],axis=2))) ,axis=2)
    F[1:,:,L3+1:] = np.flip(F[1:,:,L3+shift:],axis=0)
    F[:,1:,L3+1:] = np.flip(F[:,1:,L3+shift:],axis=1)
    output = np.fft.ifftn(F,s=L,norm='ortho')
    return np.real(output)

def BuildA(n_recs,n_shots,n_times,n_sources,dither,t_sample_rate,
    time_max,curvelet,curv,curv_size,n3_fft,n2_fft,n1_fft,L):
    np.random.seed(9389483)
    timeshifts = (np.random.rand(n_shots)*dither - dither/2)
    timeshifts[0] = 0.0
    start_time = np.linspace( 0, (n_shots-1)*time_max/n_sources,
    n_shots)
    start_time += timeshifts
    start_time = np.around(start_time/t_sample_rate).astype(int)*
    t_sample_rate
    t_blended_shot = start_time[n_shots-1]+time_max
    n_blended_times = np.int(t_blended_shot*1000)//np.int(t_sample_rate
    *1000) + 1
    elems = np.ones((n_shots*n_times))
    rows = np.zeros_like(elems)
    cols = np.zeros_like(elems)
    inc_shot = n_times
    for i in np.arange(n_shots):
        index_start = i*inc_shot
```

```
        index_end   = index_start+inc_shot
        row_start = int(start_time[i]/t_sample_rate)
        row_end   = row_start + inc_shot
        col_start = index_start
        col_end   = index_end
        rows[index_start:index_end] = np.arange(row_start,  row_end)
        cols[index_start:index_end] = np.arange(col_start,  col_end)
    A_temp = sparse.csr_matrix( (elems,(rows,cols)), shape=(
n_blended_times , n_shots*n_times) )

    def Amatvec(x_vector):
        if curvelet == True:
            data = np.real(curv.inv(x_vector))
            x_vector = data.reshape((n_recs*n_shots*n_times))
            b_vector = np.zeros((n_recs*n_blended_times))
        else:
            data = ifft3D(x_vector.reshape((n3_fft,n2_fft,n1_fft)),L)
            x_vector = data[:n_recs,:n_shots,:n_times].reshape((n_recs*
n_shots*n_times))
            b_vector = np.zeros((n_recs*n_blended_times))

        for i in np.arange(n_recs):
            x_vector_i = x_vector[i*n_shots*n_times:(i+1)*n_shots*
n_times]
            b_vector[i*n_blended_times:(i+1)*n_blended_times] = A_temp.
dot(x_vector_i)
        return b_vector

    def Armatvec(b_vector):
        x_vector      = np.zeros((n_recs*n_shots*n_times),dtype='
complex64')
        for i in np.arange(n_recs):
            b_vector_i = b_vector[i*n_blended_times:(i+1)*
n_blended_times]
            x_vector[i*n_shots*n_times:(i+1)*n_shots*n_times] = (A_temp
.getH()).dot(b_vector_i)
        data = x_vector.reshape((n_recs,n_shots,n_times))
        if curvelet == True:
            x_vector = curv.fwd(data)
        else:
            data = fft3D(data,L)
            x_vector = data.reshape((n3_fft*n2_fft*n1_fft,))
        return x_vector
    if curvelet == True:
        A = linalg.LinearOperator((n_recs*n_blended_times , curv_size),
matvec=Amatvec, rmatvec=Armatvec)
    else:
```

```
        A = linalg.LinearOperator((n_recs*n_blended_times , n3_fft*
    n2_fft*n1_fft),matvec=Amatvec, rmatvec=Armatvec)
    return A,n_blended_times

######################### MAIN ##############################
RSF_IN  = sf.Input()
RSF_OUT_BLD = sf.Output()
par = sf.Par()
n_sources = int(par.string("n_sources"))
dither = float(par.string("dither"))
curvelet = par.string("S","fft")
# Read axes
a1 = RSF_IN.axis(1); n_times = a1['n']; o1  = a1['o']; t_sample_rate =
    a1['d']  #TIMES
a2 = RSF_IN.axis(2); n_shots = a2['n']; o2  = a2['o']; d2 = a2['d'] #
    SHOTS
a3 = RSF_IN.axis(3); n_recs  = a3['n']; o3  = a3['o']; d3 = a3['d'] #
    RECEIVERS
n_times_orig = RSF_IN.int("ntimes_orig")
n_shots_orig = RSF_IN.int("nshots_orig")
time_max = (n_times_orig-1)*t_sample_rate
verbose = False
if curvelet == "curvelet":
    sys.stderr.write("Using Curvelet as Sparce transform\n")
    curvelet = True
    from pyct.fdct3 import fdct3
    nbs = 4 # number of scale for the decomposition
    nba = 16 # number of angle at the 2nd coarsest scale
    ac  = 1 # curvelets at finest scale
    curv_size = 0
    curv = fdct3((n_recs,n_shots_orig,n_times_orig),nbs,nba,ac,cpx=
    False)
    tmp = np.zeros((n_recs,n_shots_orig,n_times_orig))
    tmp = curv.fwd(tmp)
    curv_size = tmp.size
else:
    sys.stderr.write("Using FFT as Sparce transform\n")
    curvelet = False
    curv = None
    curv_size = 0
    padding = 100
    n3_fft = n_recs+padding
    n2_fft = n_shots_orig+padding
    n1_fft = (n_times_orig+padding)//2+1
    L = [n3_fft,n2_fft,n_times_orig+padding]

# Bulding matrix A
```

```
sys.stderr.write("Building matrix A\n") if verbose else verbose
A,n_blended_times = BuildA(n_recs,n_shots_orig,n_times_orig,n_sources,
    dither,t_sample_rate,time_max,curvelet,curv,curv_size,n3_fft,n2_fft,
    n1_fft,L)
sys.stderr.write("    A Shape = (%d,%d)\n" %(A.shape[0],A.shape[1])) if
    verbose else verbose
# Setting output dimensions
a2['n'] = n_shots_orig
a1['n'] = n_times_orig
RSF_OUT_BLD.putaxis(a1,1)
RSF_OUT_BLD.putaxis(a2,2)
RSF_OUT_BLD.putaxis(a3,3)

b_vector      = np.zeros((n_recs*n_blended_times))
sys.stderr.write("Reading data\n")
for i in np.arange(n_recs):
    input_data = RSF_IN.read(shape=(n_shots,n_blended_times)) #READ
   RECEIVER
    sys.stderr.write("    Input Shape = (%d,%d)\n" %(input_data.shape
    [0],input_data.shape[1])) if verbose else verbose
    sys.stderr.write("    Reshaping data to vector\n") if verbose else
    verbose
    b_vector[i*n_blended_times:(i+1)*n_blended_times] = input_data.
    reshape((n_blended_times,))
sys.stderr.write("    b_vector Shape = (%d,)\n" %(b_vector.shape[0])) if
    verbose else verbose
sys.stderr.write("Deblending data (SPGl1)\n")
if curvelet == "curvelet":
    x_vector, resid, grad, info = spgl1.spg_bpdn(A, b_vector, 0.01,
    verbosity=2,iscomplex=False)
else:
    x_vector, resid, grad, info = spgl1.spg_bpdn(A, b_vector, 0.01,
    verbosity=2,iscomplex=True)
sys.stderr.write("    x_vector Shape = (%d,)\n" %(x_vector.shape[0])) if
    verbose else verbose
sys.stderr.write("Writing data to disk\n")
if curvelet == True:
    output_data = np.real(curv.inv(x_vector))
else:
    output_data = ifft3D( x_vector.reshape((n3_fft,n2_fft,n1_fft)),L )
    output_data = output_data[:n_recs,:n_shots_orig,:n_times_orig]
for i in np.arange(n_recs):
    RSF_OUT_BLD.write(output_data[i,:,:])
RSF_OUT_BLD.close()
RSF_IN.close()
sys.exit(0)
```