SEISMIC SINGULARITY CHARACTERIZATION WITH REDUNDANT DICTIONARIES

by

CATHERINE MAREVA DUPUIS

Graduée en Ingénierie, Ecole Nationale Supérieure de Techniques Avancées, Paris, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES

(Mathematics)

THE UNIVERSITY OF BRITISH COLUMBIA

July 2005

(Signature) _____

Department of Mathematics
The University of British Columbia
Vancouver, Canada

Date _____

Abstract

We consider seismic signals as a superposition of waveforms parameterized by their fractional-orders. Each waveform models the reflection of a seismic wave at a particular transition between two lithological layers in the subsurface. The location of the waveforms in the seismic signal corresponds to the depth of the transitions in the subsurface, whereas their fractional-order constitutes a measure of the sharpness of the transitions. By considering fractional-order transitions, we generalize the zero-order transition model of the conventional deconvolution problem, and aim at capturing the different types of transitions. The goal is to delineate and characterize transitions from seismic signals by recovering the locations and fractional-orders of its corresponding waveforms. This problem has received increasing interest, and several methods have been proposed, including multi- and monoscale analysis based on Mallat's wavelet transform modulus maxima, Spice based on the continuous wavelet transform, and seismic atomic decomposition.

We propose a new method based on a two-step approach, which divides the initial problem of delineating and characterizing transitions over the whole seismic signal, into two easier subproblems. The algorithm first partitions the seismic signal into its major components, and then estimates the fractional-orders and locations of each component. Both steps are based on the sparse decomposition of seismic signals in overcomplete dictionaries of waveforms parameterized by their fractional-orders, and involve ℓ^1 minimizations solved by an iterative thresholding algorithm. We present the complete method, and show numerical results on both synthetic and real data.

Table of Contents

List of Tables

List of Figures

Acknowledgments

First, I would like to acknowledge my engineering school ENSTA (Ecole Nationale Supérieure de Techniques Avancées) in Paris, France, whose excellent program enabled to study at the University of British Columbia. I am also very grateful to Dr. Robert Braid and all my professors at ENSTA for their help and support.

I would also like to thank my supervisor, Dr. Felix Herrmann, for providing me with a very interesting project and guiding me in my research. I also thank him for enabling me to participate in the MGA (Multiscale and Geometry Analysis) program organized by IPAM (Institute for Pure and Applied Mathematics) at the University of California, Los Angeles. Doing research at IPAM for the Fall 2004 as part of the MGA program has been a wonderful experience.

I also thank Professor Ozgur Yilmaz for useful discussions and suggestions on my work, and my officemates Peyman Poor Moghaddam, Gilles Hennenfent and Urs Boeniger for their help and friendship.

I gratefully acknowledge a Jean-Walter Zellidja scholarship for my first year at UBC.

Finally, I give personal thanks to my parents for their love, support and encouragement throughout my studies.

Part I

Capita Prima

Chapter 1
Introduction

The Earth's subsurface consists of layers of different materials separated by interfaces, also called transitions. Transitions are characteristic of regions where the Earth's properties (indicated by changes in the acoustic properties) vary rapidly compared to the scale of the seismic wavelet. These sudden changes in the properties at the transitions create edges in the medium and provoke the reflection of seismic waves. These reflections are recorded at the surface, giving rise to a seismic signal (seismic trace) which contains the seismic events triggered by the reflection of the waves at the different transitions. To a good approximation, a seismic signal can be written as the convolution of the seismic reflectivity and the seismic source function (seismic wavelet) [5]. Here the reflectivity depends on the transitions in the acoustic impedance, which is the product of the compressional wave speed and the density of the materials. Seismic data thus carries information on the geometrical relationships between subsurface lithological boundaries, which is usually used to build geological models to explain the origin of stratigraphic features.

Extracting local information about transitions from seismic data has recently received increasing interest [18, 6, 29, 14], as very precise knowledge of the subsurface is more and more needed. The accurate locations and nature of transitions are very important, as they provide geologists with quantitative and complementary information that can be used in various geophysical applications, ranging from improving the geological interpretation of the subsurface, to detecting changes in the lithology. Studies by Herrmann et al. on the rock physics [15], also showed that permeability properties of the rocks in the subsurface, were linked to the nature of transitions.

They proposed a percolation model that could explain the various natures of the transitions from their permeability properties [15]. The ultimate goal is therefore to obtain the permeability of the rocks from seismic data analysis.

The diversity of transitions in the subsurface has recently been established by Herrmann et al. [10, 12] in their study of seismic and well data using multiscale analysis with wavelets. Their results showed that the Earth's subsurface behaved like a multifractal, giving rise to a medium consisting of accumulations of varying order singularities [10, 12, 14]. Transitions no longer followed the typical zero-order discontinuity model (e.g. step function), but could have any fractional-order discontinuity, that could be obtained by fractionally differentiating/integrating a zero-order discontinuity [13, 28].

The conventional deconvolution problem (spiky decon) considers seismic transitions in the subsurface as zero-order discontinuities, and aims at recovering the seismic reflectivity as a spike train signal. However, several methods have been proposed to delineate and characterize transitions. Starting in 1972 with Harms and Tackenberg [9], followed by Payton in 1977 [23], and then Taner et al. in 1979 [27], attempts were made to characterize transitions using complex trace analysis. This analysis is based on instantaneous phase behavior, where the instantaneous phase is seen as a seismic trace attribute giving information on lithological properties. The complex trace analysis approach is nevertheless noise sensitive, as it is based on the non-local Hilbert transform. The model used for transitions in the complex trace analysis is also quite simple as it does not incorporate all the different natures of transitions. In this paper, we propose a mathematical model for transitions that incorporates any instantaneous phase behavior through any fractional-order transition, as fractional differentiations/integrations yield phase rotations [6]. Our model therefore allows for any fractional-order transitions in the subsurface, leading to a broader class of reflectivity signals. The conventional deconvolution spike train reflectivity is a particular case of our model for zero-order discontinuities.

In 1999, following their multiscale studies of seismic and well data, Herrmann et al. [12, 13, 16] proposed a method to characterize the sharpness of transitions using monoscale analysis. In this method, they generalized the zero-order transitions to any fractional-order. The key of the method is to fractionally integrate/differentiate the seismic signal until the disappearance/appearance of a local maximum. The amount of integration/differentiation at the point of disappearance/appearance of the local maximum gives the estimate for the fractional-order of the transition. Although relatively successful, this method lacks robustness and is highly sensitive to noise and any phase rotation of the waveforms in the signal.

Following the monoscale analysis, Herrmann [14] proposed a parameterization of seismic signals with redundant dictionaries. In this framework the signal is decomposed in a dictionary of waveforms parameterized by their fractional-order, and represented by its coefficients. Assuming sparse reflectivity, the idea is to find the sparsest set of coefficients that represents the signal in the chosen dictionary. The sparsest representation contains a few large coefficients that pertain to the different seismic events in the signal. The problem of finding these coefficients (which corresponds to solving all ill-posed problem), was attacked using Matching Pursuit [11, 14, 21] and Basis Pursuit [2, 14], which are two algorithms that approximate the sparsest representation. The results showed the ill-posedness of the problem and the difficulties in finding the sparsest representation for complicated signals in large dictionaries. For simple configurations, both algorithms are able to find the sparsest representation, but fail when the complexity of the dictionary and/or the signal is too large. Based on the conclusions of these studies, we reformulate the problem into a more solvable approach. We propose a two-step method that divides the resolution of the problem into two subproblems, that we call detection and estimation.

By finding the largest sets of coefficients in the approximate sparse representation of the seismic signal in a chosen detection dictionary $\mathcal{D}_d$, the detection algorithm locates the major features of the seismic signal that correspond to predominant reflectors in the Earth. We use a detection dictionary $\mathcal{D}_d$ of parameterized waveforms which mimic the behavior of the seismic signal. The

approximate representation $\underline{c}$ of the signal $\underline{s}$ in $\mathcal{D}_d$ is obtained by minimizing

$$\frac{1}{2}\|\underline{s} - \mathbf{\Phi}_d \underline{c}\|_2^2 + \lambda_d \|\underline{c}\|_1, \tag{1.1}$$

with respect to $\underline{c}$, where $\underline{c}$ is the representation of the seismic signal $\underline{s}$ with respect to the dictionary $\mathcal{D}_d$, $\mathbf{\Phi}_d$ is the matrix of the detection dictionary $\mathcal{D}_d$, and λ_d is a trade-off parameter between the data misfit and the ℓ^1 norm of the coefficient vector $\underline{c}$. The different signal components associated with the largest coefficients in the representation $\underline{c}$, constitute a partitioning of the signal $\underline{s}$ into its prevailing localized components (predominant reflection events). The extraction of these components is done by multiplying the signal with localized windows centered at the different component locations. The extracted signals, now localized, are analyzed separately in the estimation part. We denote $\underline{s}_i$, the ith signal component extracted from $\underline{s}$. The detection step can be viewed as a spiky deconvolution problem, where the goal is to find a spike train signal containing the locations of the predominant components.

The estimation algorithm operates on relatively simple and localized signals $\underline{s}_i$ assumed to be a linear combination of a limited number of waveforms. The estimation process delineates the different waveforms in each signal $\underline{s}_i$, and estimates their fractional-order by finding the sparsest representation of the signal in another dictionary of parameterized waveforms noted $\mathcal{D}_e$. Each $\underline{s}_i$ is analyzed in the estimation part through the minimization of

$$\frac{1}{2}\|\underline{s}_i - \mathbf{\Phi}_e \underline{c}_i\|_2^2 + \lambda_e \|\underline{c}_i\|_1, \tag{1.2}$$

with respect to $\underline{c}_i$, where $\underline{s}_i$ denotes the ith signal component of the seismic signal $\underline{s}$ and $\underline{c}_i$ its representation with respect to the estimation dictionary $\mathcal{D}_e$. Here, we aim at an exact reconstruction, which requires the parameter λ_e to be very small. When $\lambda_e \to 0$, (1.1) behaves like Basis Pursuit [2] and translates into

$$\min_{\underline{c}_i} \|\underline{c}_i\|_1 \text{ subject to } \underline{s}_i = \mathbf{\Phi}_e \underline{c}_i. \tag{1.3}$$

The coefficients in the sparse representation $\underline{c}_i$ correspond to waveforms in $\mathcal{D}_e$ whose parameter α constitute estimates for the fractional-orders of $\underline{s}_i$. The final result groups the different

waveforms and associated fractional-orders pertaining to each signal component $\underline{s}_i$, in a global vector describing the complete seismic signal $\underline{s}$. The global vector contains the fractional-orders of all the reflection events of the seismic signal $\underline{s}$. The following figure shows the main steps of our detection-estimation method.

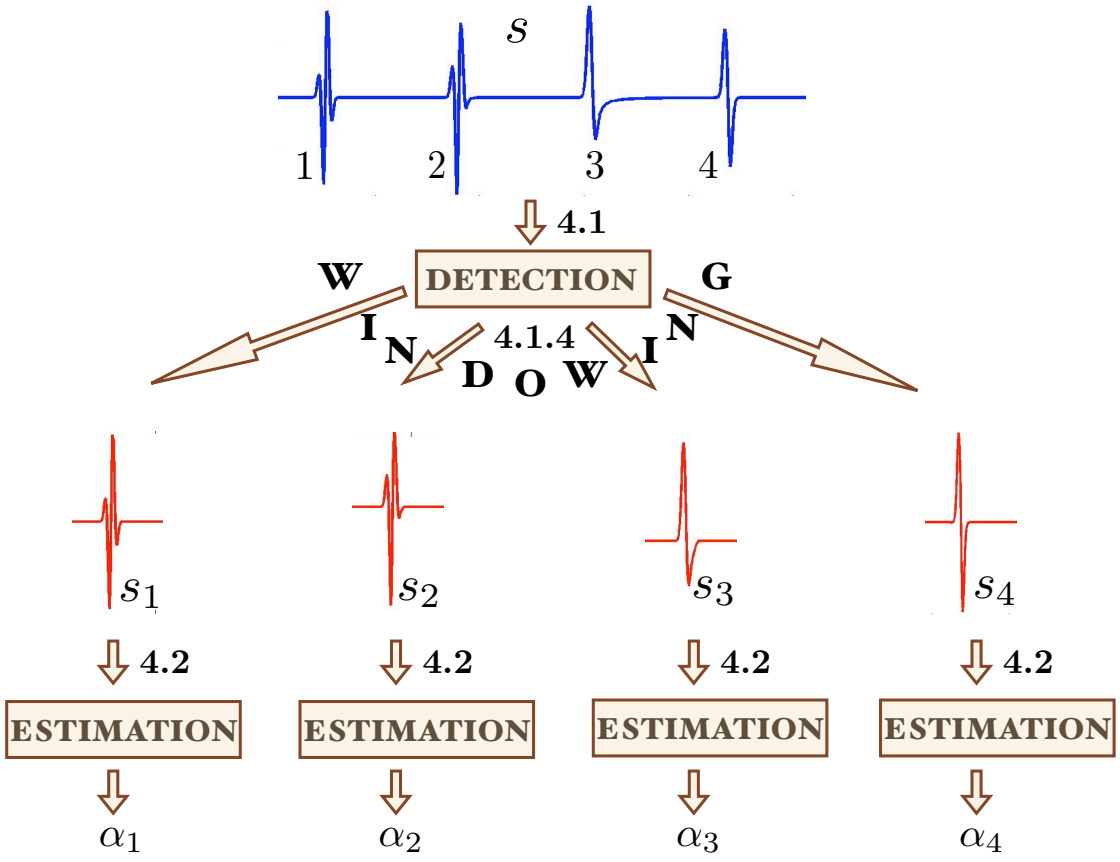

Figure 1.1: Detection-estimation method. The numbers 4.1, 4.1.4 and 4.2 refer to the sections where the particular topic is explained in detail.

Chapter 2
Background

2.1 Model for seismic transitions

Significant variations in the Earth's properties over the length-scale of the seismic wavelet create singularities in the subsurface and make seismic waves reflect. These singularities occur at transitions between layers, and are typically modeled by zero-order discontinuities (i.e. step discontinuities). However, multiscale analysis on sedimentary records (vertical profiles of the broadband fluctuation of the elastic properties) performed by Herrmann et al. [10, 12, 22], revealed the existence of accumulations of varying order singularities in the subsurface, which give rise to any fractional-order discontinuity. The different types of transitions correspond to the different materials in the Earth and to the various ways in which materials are deposited and mixed together at the transitions.

2.1.1 The seismic reflectivity

The mathematical model

Following Herrmann et al. [17, 11, 13, 14], we extend the usual zero-order transition model to any fractional-order transition, modeled by the causal and anti-causal fractional splines [28] (also called fractional onset functions):

$$\text{causal: } \chi_+^\alpha(x) = \begin{cases} 0 & \text{if } x < 0 \\ \frac{x^\alpha}{\Gamma(\alpha+1)} & \text{if } x \geq 0 \end{cases}, \qquad \text{and anti-causal: } \chi_-^\alpha(x) = \begin{cases} \frac{(-x)^\alpha}{\Gamma(\alpha+1)} & \text{if } x \leq 0 \\ 0 & \text{if } x > 0, \end{cases} \tag{2.1}$$

where $\alpha \geq 0$ and Γ is the Gamma function defined as: $\Gamma(x) = \int_0^\infty t^x e^{-t} dt$ for $x \in \mathbb{R}$. When $\alpha \geq 1$, these functions are differentiable. When α is between 0 and 1, they become non-differentiable but remain continuous. For $-1 < \alpha \leq 0$, they are discontinuous (and non-differentiable). When $\alpha \leq -1$, the fractional splines cease to be locally integrable, and become tempered distributions [13]. Causal and anti-causal fractional splines are linked together by the relationship: $\forall x \in \mathbb{R}$, $\chi_-^\alpha(x) = \chi_+^\alpha(-x)$. Mathematically, the fractional exponents α specify the regularity of the fractional splines, as they coincide with their Lipschitz exponent. Each of these fractional splines [28] is Lipschitz α, where the Lipschitz regularity is defined as:

Definition 2.1.1 (Lipschitz regularity [20])

- *A function f is pointwise Lipschitz $\alpha \geq 0$ at ν if*
  $\exists K > 0$ *and a polynomial p_ν of degree $m = \lfloor \alpha \rfloor$ such that*

$$\forall t \in \mathbb{R},\ |f(t) - p_\nu(t)| \leq K|t - \nu|^\alpha.$$

- *A function f is uniformly Lipschitz α over [a,b] if it satisfies the above inequality for all $\nu \in [a, b]$, with a constant K that is independent of ν.*

- *The Lipschitz regularity of f at ν or over $[a, b]$ is the supremum of the α such that f is Lipschitz α.*

Our choice of fractional splines is motivated by their regularity and scale invariance. Thus we can characterize the sharpness of a transition *irrespectively of the scale*:

$\forall x \in \mathbb{R}$ and $\forall \alpha \geq 0, \sigma > 0$, $\chi_\pm^\alpha(\sigma x) = \sigma^\alpha \chi_\pm^\alpha(x)$.

As explained before, each of these functions has a particular order of singularity α. For $\alpha \geq 1$ at ν, the function $\chi_\pm^\alpha$ has α bounded derivatives and a singularity at ν with Lipschitz exponent α. However, when $\alpha < 1$ at ν, $\chi_\pm^\alpha$ is not differentiable at ν and α characterizes the type of singularity.

Examples of causal and anti-causal fractional splines are shown in Figures 2.1 and Figure **??**:

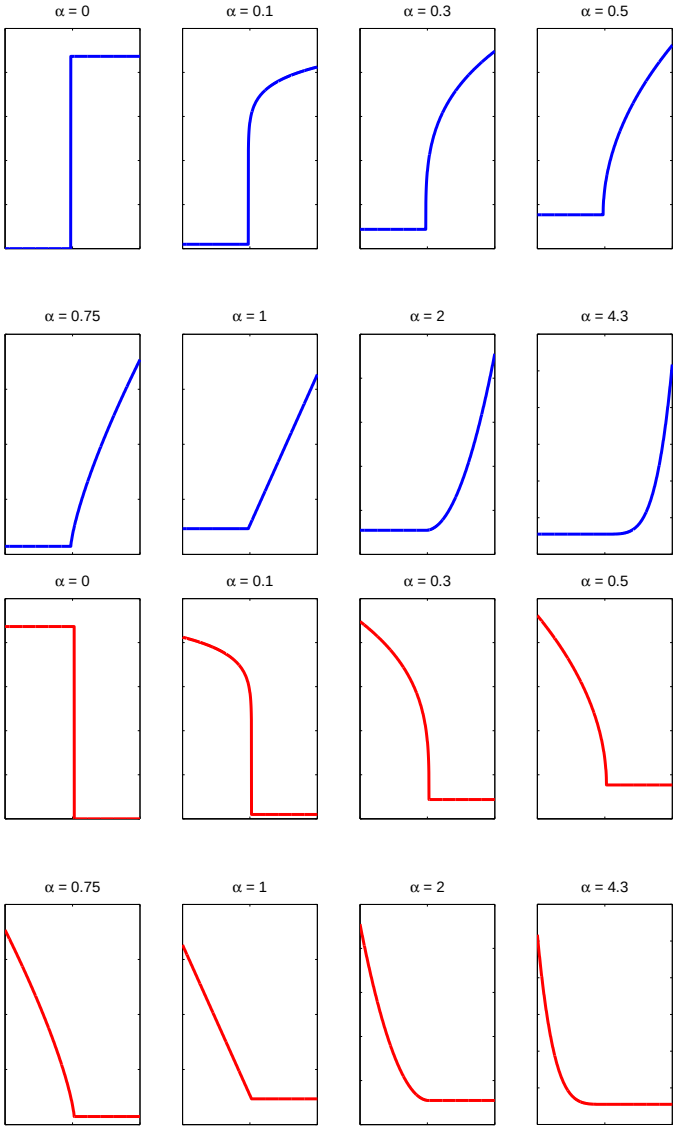

Figure 2.1: Examples of Causal and anti-causal fractional splines with different fractional-orders. The top two rows show causal fractional splines, and the bottom two rows show anti-causal fractional splines

Implementation of the $\chi_{\pm}^{\alpha}$ basis

We construct the basis $\mathcal{B}_{\alpha}^{\pm}$, made of the family of all translations of $\chi_{\pm}^{\alpha}$ for each α. Except for $\alpha = 0$, the family $\mathcal{B}_{\alpha}^{\pm}$, obtained by all the translations of $\chi_{\pm}^{\alpha}$, gives a degenerate collection of waveforms as the last basis function is zero everywhere. This is due to the fact that the first

sample of $\chi_{\pm}^{\alpha}$ is zero for any α except $\alpha = 0$. By adding 1 to all the $\chi_{\pm}^{\alpha}$, we ensure all the basis functions are non-zero. Here the translations are non circular. If we use circular translations (or circular permutations), we create a singularity of order 0 for all the basis functions of all the bases $\mathcal{B}_{\alpha}^{\pm}$. A circular translation $\mathcal{C}^p$, $p \in \mathbb{Z}$, of a vector $\underline{v} \in \mathbb{R}^N$ has the property that

$$\forall 1 \leq n \leq N \ , \ \mathcal{C}^p \underline{v}(n) = \begin{cases} v((n+p) \pmod N) & \text{if } n+p \not\equiv 0 \pmod N \\ v(N) & \text{if } n+p \equiv 0 \pmod N \end{cases}$$

Let N be the length of the the vector χ_{α}^{+} and define:

$$\mathcal{B}_{\alpha}^{+} = \left\{ v_i \right\}_{1 \leq i \leq N} \text{ , such that } \forall 1 \leq n \leq N \ , \ v_i(n) = \chi_{+}^{\alpha}(n-i+1) \ \text{, where } \chi_{+}^{\alpha}(n) = 0 \ \forall n \leq 0$$

It is easy to prove that for any $\alpha \geq 0$, $\mathcal{B}_{\alpha}^{+}$ is a basis. See proof in Appendix B.1. A similar argument is used to show that $\mathcal{B}_{\alpha}^{-}$ is a basis.

Transitions and reflectivity

We denote by f, the 1D vertical earth model that represents the transitions as a superposition of fractional splines:

$$f(z) = \sum_i a_i \chi_{\pm}^{\alpha_i}(z - z_i), \tag{2.2}$$

where z corresponds to the depth in the subsurface, and z_i the location of the ith transition in the subsurface. By differentiating f, given by (2.2), we obtain the seismic reflectivity r, as

$$r(z) = \sum_{i \in \Lambda_C} K_i \chi_{+}^{\alpha_i - 1}(z - z_i) - \sum_{i \in \Lambda_A} K_i \chi_{-}^{\alpha_i - 1}(z - z_i), \tag{2.3}$$

where $K_i = \frac{\alpha_i a_i \Gamma(\alpha_i)}{\Gamma(\alpha_i + 1)}$, and Λ_C and Λ_A are the sets of indices for the transitions with causal and anti-causal discontinuities respectively.

2.1.2 The seismic source function

In most geophysical surveys, and especially those using explosive sources, the seismic source function ψ is unknown, and is typically modeled by the Ricker/Mexican hat wavelet (opposite

sign of the second derivative of the Gaussian). In the rest of the paper, we will only use the name Ricker wavelet. We relax this model and discuss our assumptions on the seismic source function, with regard to its smoothness and wiggliness. We first introduce some definitions. Following Unser et al. [28], a γ-order causal fractional B-spline, ζ_+^γ, is defined by taking the $(\gamma+1)^{\text{th}}$ fractional difference of a one-sided power function

$$\zeta_+^\gamma(x) = \triangle_+^{\gamma+1}\chi_+^\gamma(x) = \sum_{k\geq 0}(-1)^k\binom{\gamma+1}{k}\chi_+^\gamma(x-k), \tag{2.4}$$

where $\gamma > -1$, $x \in \mathbb{R}$, and $\binom{\gamma+1}{k}$ is the generalization of the binomial coefficients defined as $\binom{\gamma+1}{k} = \dfrac{\Gamma(\gamma+2)}{\Gamma(k+1)\Gamma(\gamma+2-k)}$ [28]. The $(\gamma+1)^{\text{th}}$ difference of a function f is defined by $\triangle_+^{\gamma+1}f(x) = \sum_{k\geq 0}(-1)^k\binom{\gamma+1}{k}f(x-k)$. In the Fourier domain, (2.4) translates to

$$\hat{\zeta}_+^\gamma(\omega) = \left(\frac{1-\mathrm{e}^{-i\omega}}{i\omega}\right)^{\gamma+1}, \tag{2.5}$$

with the frequency $\omega \in \mathbb{R}$. The anti-causal and symmetric fractional splines [28] are defined in a similar way by

$$\text{anti-causal: } \hat{\zeta}_-^\gamma(\omega) = \hat{\zeta}_+^\gamma(-\omega) \text{ , and symmetric: } \hat{\zeta}_*^\gamma(\omega) = \left|\frac{sin(\omega/2)}{\omega/2}\right|^{\gamma+1}. \tag{2.6}$$

The fractional B-splines are the extensions of the traditional B-splines to non integer exponents α. For further extensions, Blu et al. [1] defined the (γ,τ)-fractional B-splines

$$\hat{\zeta}_\tau^\gamma(\omega) = \hat{\zeta}_-^{\frac{\gamma+1}{2}-\tau}(\omega)\hat{\zeta}_+^{\frac{\gamma+1}{2}+\tau}(\omega) \tag{2.7}$$

$$= \left(\frac{\mathrm{e}^{i\omega}-1}{i\omega}\right)^{\frac{\gamma+1}{2}-\tau}\left(\frac{1-\mathrm{e}^{-i\omega}}{i\omega}\right)^{\frac{\gamma+1}{2}+\tau}, \tag{2.8}$$

where α is the usual fractional exponent, and τ controls the phase of the function. Unlike the traditional B-splines, fractional splines do not have compact support, but are of rapid decay with the following decay property [28]

$$\zeta^\gamma(x) \underset{x\to\infty}{=} \mathcal{O}\left(\frac{1}{x^{\gamma+1}}\right). \tag{2.9}$$

The fractional B splines and $(\gamma, 0)$-fractional B-splines converge to the infinitely smooth Gaussian function when $\gamma \to \infty$. We consider the source function as a β^{th} derivative of an (γ, τ)-fractional B-spline, parameterized by its fractional exponent s α and β, and its phase determined by τ

$$\psi(t) = \partial^{\beta} \zeta_{\tau}^{\gamma}(t) \text{ for } \beta < \gamma + 1, \tag{2.10}$$

where t is the time variable. The conventional Ricker wavelet model for the seismic source function ψ is a particular case of our model for $\beta = 2$, $\tau = 0$, and $\gamma \to \infty$. In addition to this model, we discuss two properties of the seismic source function ψ: the decay of its Fourier spectrum for frequencies going to infinity, as $\psi \in \mathcal{C}^{\gamma}(\mathbb{R})$ (ψ is an α-times continuously differentiable function on $\mathbb{R}$),

$$\int_{\mathbb{R}} |\hat{\psi}(\omega)||\omega|^{\gamma} d\omega < \infty,$$

and the differentiability of the Fourier transform of ψ at zero frequency [14]:

$$\int_{\mathbb{R}} t^{q} \psi(t) dt = 0 \Longleftrightarrow \mathcal{R}(\partial_{\omega}^{q} \hat{\psi})_{\omega=0} = 0 \text{ for } q \in \mathbb{N}, \text{ and } q \leq M.$$

The first condition limits the high-frequency content of ψ by setting the asymptotic decay rate for high frequencies, and the second condition requires that ψ be orthogonal to some finite degree polynomial, hence describing its wiggliness [14]. This latter property defines the number of vanishing moments of ψ [20], noted q.

2.1.3 The model for the imaged seismic signal

We consider the linearized single scattering approximation, where the imaged seismic signal s can be written as:

$$s(z) \propto (r * \psi_z)(z). \tag{2.11}$$

Here, r is the imaged travel-time depth converted reflectivity, and ψ_z is the depth converted seismic source function. For clarity purposes, we simplify the notations and refer to the depth converted seismic source function ψ_z as ψ. Because we are only interested in the singularity

properties of the seismic signal, we simplify the notations and omit the constants. We therefore write the seismic signal s as the convolution of the reflectivity and the seismic source function:

$$\begin{aligned} s(z) &= (r * \psi)(z) && \text{(2.12a)} \\ &= \sum_{i \in \Lambda_C} K_i \chi_+^{\alpha_i - 1}(z - z_i) * \psi(z) - \sum_{i \in \Lambda_A} K_i \chi_-^{\alpha_i - 1}(z - z_i) * \psi(z), && \text{(2.12b)} \end{aligned}$$

where r is given by equation (2.3). The seismic signal s can also be written as a superposition of fractional derivatives/integrations of the seismic wavelet ψ, where the definition of fractional differentiation is given by Liouville [19] in his generalization of differentiation for fractional-orders

Definition 2.1.2 (Fractional Derivatives [19]) *Let f be a tempered distribution in $\mathcal{S}(\mathbb{R})$ and $\chi_+^{\alpha-1}$ be the causal fractional spline:*

The fractional derivative of f of order α is defined as

$$D^\alpha f = \chi_+^{-\alpha-1} * f,$$

where the convolution has to be taken in the sense of distributions.

Under the condition that the orders α_i given in equation (2.12b) are all non integers, and that the seismic wavelet ψ is either even or odd, we can write:

$$\chi_+^{\alpha_i - 1}(z - z_i) * \psi(z) = C_i D^{-\alpha_i} \psi(z - z_i) \tag{2.13}$$

and

$$\chi_-^{\alpha_i - 1}(z - z_i) * \psi(z) = \begin{cases} C_i D^{-\alpha_i} \psi(z_i - z) \text{ if } \psi \text{ is odd,} \\ -C_i D^{-\alpha_i} \psi(z_i - z) \text{ if } \psi \text{ is even,} \end{cases} \tag{2.14}$$

which gives a seismic signal s of the form

$$s(z) = \sum_{i \in \Lambda_C} C_i D^{-\alpha_i} \psi(z - z_i) \pm \sum_{i \in \Lambda_A} C_{\alpha_i} D^{-\alpha_i} \psi(z - z_i), \tag{2.15}$$

where $C_i = \frac{\Gamma(-\alpha_i)K_i}{\Gamma(\alpha_i)} = \frac{\alpha_i a_i \Gamma(-\alpha_i)}{\Gamma(\alpha_i+1)}$.

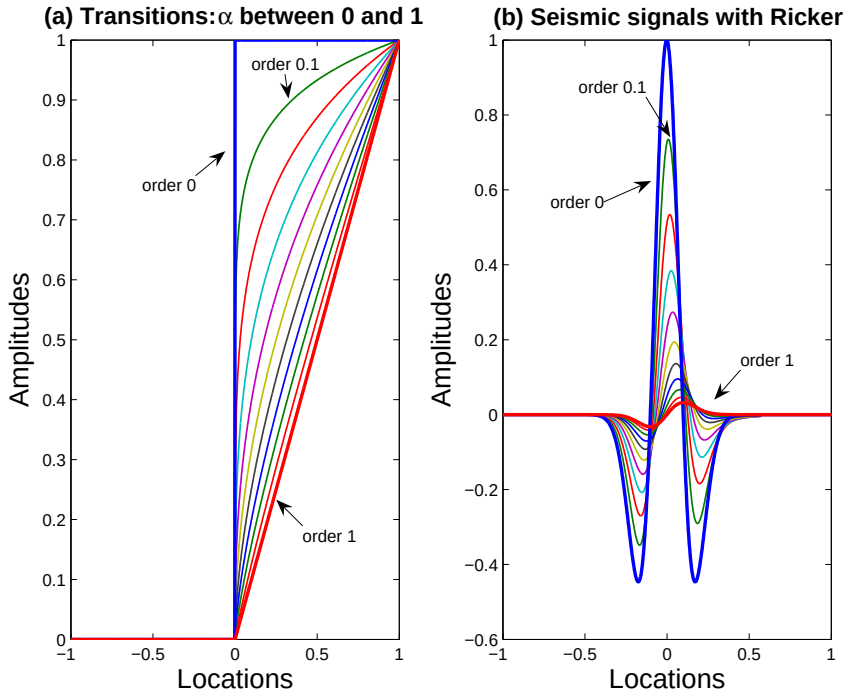

Figure 2.2: Varying order transitions and their associated seismic signals obtained with a Ricker wavelet. The transitions are set to continuously interpolate between a zero-order discontinuity (step function) and a first-order discontinuity (ramp function). One can clearly see that different order transitions create seismic signals that not only differ in amplitudes, but also in waveform.

In Figure 2.2, we present an example of varying order transitions and their corresponding seismic signals. The seismic source function is a Ricker wavelet. Figure 2.2a shows the different transitions with fractional-orders ranging from 0 to 1 and Figure 2.2b shows the seismic signals obtained by convolving the reflectivity associated to these transitions and the Ricker wavelet. By interpolating the fractional-order of the transitions between 0 and 1, we obtain seismic signals going from the Ricker wavelet (associated to the zero-order transition) to the derivative of the Ricker wavelet (associated to the first-order transition).

2.2 Previous work

2.2.1 Complex seismic trace analysis

Beginning with Harms and Tackenberg [9] in 1972, followed by Payton in 1977 [23], and later Taner et al. [27] in 1977 and 1979, many attempts have been made to characterize transitions using complex trace analysis. Complex seismic trace analysis calculates attributes from a seismic trace. Attributes are defined as specific information characterizing seismic traces that can be computed or extracted from them. For complex trace analysis, the attributes are the amplitude, phase, instantaneous frequency, weighted average frequency and apparent polarity [27]. This analysis considers a seismic trace $\underline{s}(t)$ as the real part of a complex signal $S(t)$ [27], where S is defined as:

$$S(t) = s(t) + is^*(t) \ , \forall t \geq 0.$$

s^* is called the quadrature component and is uniquely determined by the following two conditions

1. s^* is calculated from $\underline{s}$ by a convolution
2. If $s(t) = Acos(\omega t + \theta)$ for $A, \theta \in \mathbb{R}$ and $\omega > 0$, then $s^*(t) = Asin(\omega t + \theta)$ (s^* is the Hilbert transform of s)

$\underline{s}$, considered as a complex signal, can be written as $S(t) = A(t)e^{i\theta(t)}$, where $s(t)$ is the real part of S, and $s^*(t)$ the imaginary part. From this, we refer to the amplitude $|A(t)|$ as the *reflection strength*, and to the phase $\theta(t)$ as the *instantaneous phase* [27]. The instantaneous frequency ω can be computed by calculating:

$$\omega(t) = \frac{d\theta(t)}{dt}.$$

Using the reflection strength, the instantaneous phase and instantaneous frequency, we deduce the weighted average frequency as [27]:

- Weighted average frequency:

$$\bar{\omega}(t) = \frac{\int_{-\infty}^{\infty} A(t-\tau)\omega(t-\tau)L(\tau)d\tau}{\int_{-\infty}^{\infty} A(t-\tau)L(\tau)d\tau}$$

- Apparent polarity

  The apparent polarity is defined as the sign of $s(t)$ when $A(t)$ has a local maximum [27].

Taner et al. [27] interpreted these attributes as information on geological and lithological features. For instance, they interpreted high reflection strength as "major lithological changes between adjacent rock layers, such as uncomformities and boundaries associated with sharp changes in sea level or depositional environments", and showed that "phase displays are effective in showing discontinuities ... and events with different dip attitudes which interfere with each other" [27].

This analysis was based on instantaneous phase behavior, where the instantaneous phase was seen as a seismic trace attribute giving information on lithological properties. The complex trace analysis approach is noise sensitive and has the disadvantage of being based on the non-local Hilbert transform. The model they used for transitions was perhaps too simplistic. In this paper, we propose a mathematical model for transitions that any incorporates instantaneous phase behavior of the form $(i\omega)^\alpha = \mathrm{e}^{i\frac{\pi}{2}\alpha}$, through any fractional-order transition, as fractional differentiations/integrations yield phase rotations [6].

2.2.2 Multiscale analysis

Herrmann et al. [10, 12, 17], performed a multiscale analysis of seismic data using wavelets.

Theorems and definitions

Given a mother wavelet ψ, where ψ is a normalized function in $L^2(\mathbb{R})$ with zero average, we can create a family of wavelets through translations and dilations of the mother wavelet

$$\forall t \in \mathbb{R}\ ,\ \forall (\alpha, \sigma) \in (\mathbb{R}^+)^2 \psi_{a,\sigma}(t) = \frac{1}{\sqrt{\sigma}} \psi\left(\frac{t-a}{\sigma}\right).$$

where $a \geq 0$ is the position of the wavelets and $\sigma \geq 0$ their scale. The wavelet coefficients of a

function f are computed by taking the inner product of f with the different wavelets $\psi_{a,\sigma}$:

$$\mathcal{W}\{f,\psi\}(a,\sigma) = \langle f, \psi_{a,\sigma} \rangle \tag{2.16}$$

See Appendix A for more details on wavelets.

Theorem 2.2.1 *[JAFFARD] [20] [Relation between Lipschitz exponent and the decay rate of the wavelet coefficients]*

If $f \in L^2(\mathbb{R})$ is Lipschitz $\alpha \leq n$ at ν, then there exists A such that:

$$\forall (u,s) \in \mathbb{R} \times \mathbb{R}^+ \ , \ |Wf(u,s)| \leq A s^{\alpha+1/2}(1 + |\frac{u-\nu}{s}|^{\alpha})$$

Conversely, if $\alpha < n$ is not an integer and there exist A and $\alpha' < \alpha$ such that

$$\forall (u,s) \in \mathbb{R} \times \mathbb{R}^+ \ , \ |Wf(u,s)| \leq A s^{\alpha+1/2}(1 + |\frac{u-\nu}{s}|^{\alpha'})$$

then f is Lipschitz α at ν

This theorem proves that the local Lipschitz regularity of f at ν depends on the decay rate at fine scales of $|Wf(u,s)|$ in the neighborhood of ν and that the decay can be controlled from its local maximal values. [20]

Definition 2.2.1 *[Modulus Maximum and Maxima Lines] A modulus maximum is a point (u_0, s_0) where $|Wf(u,s_0)|$ is maximum at $u = u_0$. This implies*

$$\frac{\partial Wf(u_0,s_0)}{\partial u} = 0$$

This local maximum should be a strict local maximum in either the right or the left neighborhood of u_0, to avoid having any local maxima when $|Wf(u,s_0)|$ is constant. We define a maxima line to be any connected curve $s(u)$ in the scale-space plane (u,s) along which all points are modulus maxima.

The Wavelet Transform Modulus Maxima Lines are referred to as WTMML. Singularities are detected by finding the abscissa where the wavelet modulus maxima converge at fine scale i.e., when $s \to 0$.

Definition 2.2.2 (Self-similarity [20]) *A set $S \subset \mathbb{R}^n$ is said to be self-similar if it is the union of disjoint subsets $S_1, S_2, ..., S_k$ that can be obtained from S by a scaling, translation and rotation. This self-similarity often implies an infinite multiplication of details, that creates irregular structures.*

Definition 2.2.3 (Capacity dimension [20]) *The capacity dimension is a simplification of the Hausdorff dimension that is easier to compute numerically. Let $\mathcal{S}$ be a bounded set in $\mathbb{R}^n$. We count the minimum number $\mathcal{N}(s)$ of balls of radius s required to cover $\mathcal{S}$. If $\mathcal{S}$ is a set of dimension D with a finite length(D=1), surface(D=2) or volume(D=3) then:*

$$N(s) \sim \frac{1}{s^D}$$

so

$$D = -\lim_{s \to 0} \frac{\log N(s)}{\log s}$$

The capacity dimension D of $\mathcal{S}$ generalizes this result and is defined by:

$$D = \liminf_{s \to 0} \frac{\log N(s)}{\log s}$$

The Hausdorff dimension is a refined fractal measure that considers all covers of $\mathcal{S}$ with balls of radius smaller than s. It is most often equal to the capacity dimension, but not always.

Definition 2.2.4 *[Singularity Spectrum [20]] Let S_α be the set of all the points $t \in \mathbb{R}$ where the pointwise Lipschitz regularity of f is equal to α. The spectrum of singularity $D(\alpha)$ of f is the fractal dimension of S_α. The support of $D(\alpha)$ is the set of α such that S_α is not empty.*

The singularity spectrum gives the proportion of Lipschitz α singularities that appear at any scale s. A multifractal is said to be homogeneous if all singularities have the same Lipschitz exponent α_0, which means that the support of $D(\alpha)$ is restricted to $\{\alpha_0\}$.

Definition 2.2.5 *[Partition function [20]] Let $\{u_p(s)\}_{p\in Z}$ be the position of all local maxima of $|W_g(u,s)|$ at a fixed scale s. The partition function $\mathcal{Z}$ measures the sum at a power q of all these wavelet modulus maxima:*

$$\mathcal{Z}(q,s) = \sum_p |Wf(u_p,s)|^q$$

For each $q \in \mathbb{R}$, the scaling exponent $\tau(q)$ measures the asymptotic decay of $\mathcal{Z}(q,s)$ at fine scales s:

$$\tau(q) = \lim_{s\to 0} inf \frac{\log(\mathcal{Z}(q,s))}{\log s}$$

This typically means that $\mathcal{Z}(q,s) \sim s^{\tau(q)}$.

Theorem 2.2.2 *[ARNEODO, BACRY, JAFFARD, MUZY] [20] Let $\Lambda = [\alpha_{min}, \alpha_{max}]$ be the support of $D(\alpha)$. Let ψ be a wavelet with $n > \alpha_{max}$ vanishing moments. If f is a self-similar signal then:*

$$\tau(q) = \min_{\alpha\in\Lambda}(q(\alpha + 1/2) - D(\alpha)) \text{ (Legendre transform)}$$

Proposition 2.2.1

- *The scaling exponent $\tau(q)$ is a convex and increasing function of q.*
- *The Legendre transform is invertible if and only if $D(\alpha)$ is convex, in which case:*

$$D(\alpha) = \min_{q\in\mathbb{R}}(q(\alpha + 1/2) - \tau(q))$$

The spectrum $D(\alpha)$ of self-similar signals is convex.

We have the relations:

$$\frac{\partial\tau}{\partial q} = \alpha(q) + 1/2$$

The $1/2$ is due to the L^2 normalization of the wavelets.

Numerical calculations [20]

1. *Maxima* Compute $Wf(u,s)$ and the modulus maxima at each scale s. Chain the wavelet maxima across scales.

2. *Partition function* Compute

$$\mathcal{Z}(q,s) = \sum_p |Wf(u_p,s)|^q$$

3. *Scaling* Compute $\tau(q)$ with a linear regression of $\log_2 \mathcal{Z}(q,s) \approx \tau(q)\log_2 s + C(q)$

4. *Spectrum* Compute

$$D(\alpha) = \min_{q\in\mathbb{R}}(q(\alpha+1/2) - \tau(q))$$

Local analysis

Herrmann [10] applied the result of theorem 2.2.1 to well data, and was able to find their local exponents α by estimating the slope of the wavelet coefficients in the log-log plane. In the log-log plane, the relationship between the wavelet coefficients and the local exponent α at position a and scale σ is:

$$\log(|\mathcal{W}\{a,\sigma\}|) \leq \log(A) + \alpha\log(\sigma), \tag{2.17}$$

where A is a positive constant. From equation (2.17), the estimate for the local exponent α is computed via linear regression. In Figure 2.3 and Figure 2.4, we show the local multiscale analysis performed on a simple synthetic signal and on a sonic well-log measurement [12]. In Figure 2.3, the top plot shows the data with the different singularities, the middle plot shows the continuous wavelet transform with the WTMML superimposed, and the bottom plots show the local multiscale analysis associated to each singularity. In Figure 2.4, the top plot shows the sonic well-log, the middle plot shows the continuous wavelet transform with the WTMML superimposed, and the bottom plots present the local multiscale analysis for selected singularities. We can clearly see that the well-log contains singularities everywhere, as the WTMML converge everywhere. This property therefore shows that the Earth behaves like a multifractal.

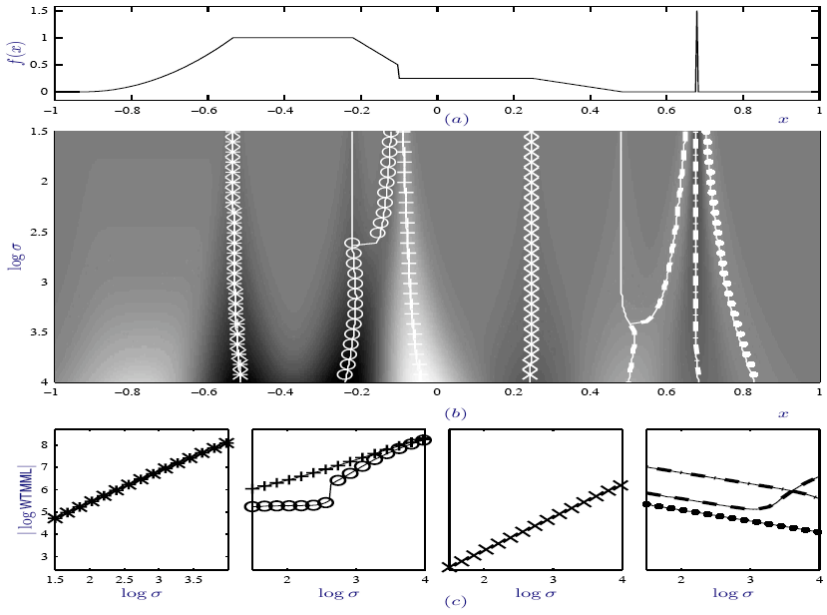

Figure 2.3: Regularity analysis through local analysis. The top plot shows the data with different singularities. The second plot shows the continuous wavelet transform with the location of the WTMM Lines superimposed. The bottom plots show the local multiscale analysis for each singularity. The slope of these different lines give an estimate for the singlarity order. Taken from [12]

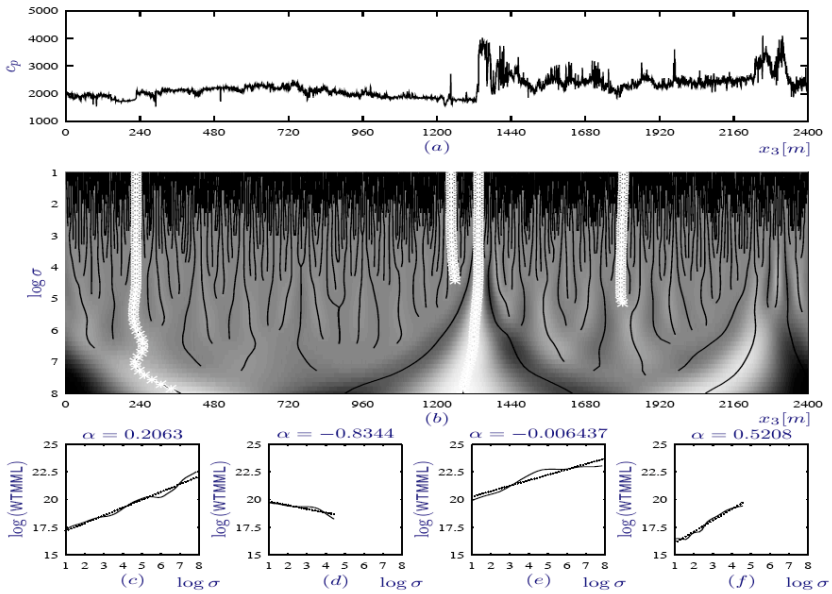

Figure 2.4: Local analysis of a sonic well-log measurement. The top plot shows the sonic well-log (compressional wave speed). The middle plot shows the continuous wavelet transform with the WTMML superimposed. The bottom plot presents the local multiscale analysis for selected singularities. We can clearly see that the well-logs are singular everywhere, and therefore present multifractal characteristics. Taken from [12]

The method is appropriate for functions with isolated singularities, because it is based on a local analysis . However, for functions with accumulated singularities (which is the case for seismic data), it is difficult to detect and characterize the singularities using a local analysis.

Global analysis

Global analysis yields an estimate for the singularity spectrum $f(\alpha)$ (defined in definition 2.2.4), which gives the "rate of occurrence" [10] of a particular singularity α in the analyzed signal. Theorem 2.2.2 and proposition 2.2.1 link the singularity spectrum $f(\alpha)$ to the scaling exponent $\tau(q)$ (defined in definition 2.2.5) through the following relation

$$\tau(q) = \min_{\alpha}(q(\alpha + 1/2) - f(\alpha)) \text{ and } f(\alpha) = \min_{q \in \mathbb{R}}(q(\alpha + 1/2) - \tau(q)).$$

"$f(\alpha)$ and $\tau(q)$ contain information on the global integrability and differentiability of the signal s " [10]. The minimum exponent α_{min} refers to the strongest singularity associated to the less regular regions, whereas α_{max} refers to the weakest singularity associated to the most regular regions. Figure 2.5 [12] shows the result of the global multiscale analysis on three different well-logs associated to the compressional wave speed, the shear wave speed and the density. The top three plot present the well-logs, the bottomm left plot shows the function $\tau(q)$ for all three logs, and the bottom right plot shows the singularity spectra $f(\alpha)$ for the three logs as well. Note the similarity between the three logs.

The results of Herrmann [10] after applying the global analysis on seismic data showed that the scale exponent α was different at every point, which is one of the characteristics of multifractals. One of the explanation for the multifractality of seismic data was given by Herrmann [10]: "The multifractality of the Earth's sedimentary deposits can be understood because the sedimentation process if closely linked to the hydrodynamics of the atmosphere. Hydrodynamic turbulence is one of the classical examples of a process showing evidence of multifractal behavior across a large inertial scale range. Hence the variability in the sedimentary deposits can be considered as a frozen state of *turbulence*." The results of the global multiscale analysis on

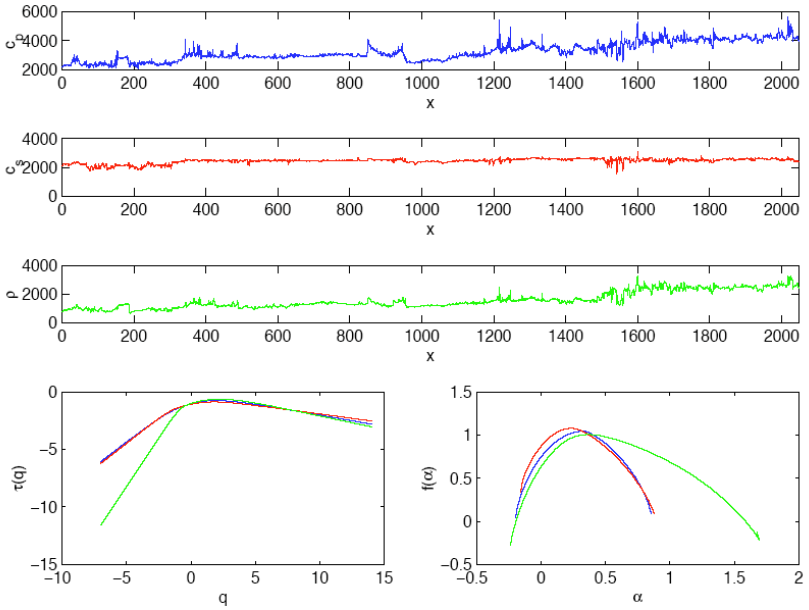

Figure 2.5: Comparison between three different well-logs. The top plot presents the compressional wave speed well-log, the second plot the shear wave well-log, and the third plot the density well-log. The bottom left figure shows the $\tau(q)$ function for all three logs, and the bottom right plot shows the singularity spectra $f(\alpha)$. Taken from [12]

seismic data provide the geophysicist with information on its global scaling exponents, but not about its localization in the data. In full band data, it is possible to estimate the local exponent α from $f(\alpha)$. However, because seismic data is bandwidth limited, this is more difficult. Indeed, the multiscale method needs all the scales (or frequencies) in order to derive the asymptotic results, which are not available in bandwidth limited data. Multiscale analysis is therefore not the best method to obtain information about the local scale exponent, albeit several attempts have been made [29, 10].

2.2.3 Monoscale analysis

Herrmann et al. [10, 12, 16] proposed a monoscale analysis method to analyze seismic data and unravel its singularities. In this method, they generalized the zero-order transitions to

any fractional-order. The key of the method was to fractionally integrate/differentiate the seismic signal until the disappearance/appearance of a local maximum. The amount of integration/differentiation at the point of disappearance/appearance of the local maximum gave the estimate for the fractional-order of the transition.

The monoscale analysis method was based on the generalization of the standard continuous wavelet transform. The continuous wavelet transform of f, defined in equation (2.16), can be rewritten as a function of the scale σ and differentiability M [12]:

$$\mathcal{W}\{f, \psi_\sigma^M\}(\sigma, a) = \sigma^M \frac{d^M}{da^M}(f * \phi_\sigma)(a) = \sigma^M (f * \frac{d^M}{da^M}\phi_\sigma)(a) = (f * \psi_\sigma^M)(a), \tag{2.18}$$

where ϕ_σ is a $2M$ differentiable, real and symmetric smoothing function with support proportional to σ, and ψ_σ^M is the wavelet generated by dilations of ψ^M defined as $\forall x \in \mathbb{R}$, $\psi^M(x) = (-1)^M \frac{d^M}{dx^M}\phi(x)$ [12].

Herrmann et al. [10, 12, 16] generalized the wavelet transform to any fractional-order derivatives β:

$$\mathcal{W}\{f, \psi_\sigma^\beta\}(\sigma, a) = \sigma^\beta \frac{d^\beta}{da^\beta}(f * \phi_\sigma)(a), \tag{2.19}$$

with $\beta \in \mathbb{R}$. When β is negative, $\frac{d^\beta}{da^\beta}$ is a fractional integration, and when β is positive, $\frac{d^\beta}{da^\beta}$ is a fractional differentiation [12, 16, 17]. Instead of varying the scale σ like the multiscale method, β is the varying parameter, changing the fractional number of differentiations or integrations [12]. Modulus Maxima lines are defined in the same way as for the standard wavelet transform, defined in definition 2.2.1.

For fractional differentiation ($\beta \geq 0$), the local estimate for the fractional exponent α is given by the minimum value of β for which a local maximum appears along these β maxima lines:

$$\hat{\alpha}(\sigma, a) = \inf_\beta \{\partial_a |\mathcal{W}\{f, \psi_\beta\}(\sigma, a)| = 0\}. \tag{2.20}$$

For fractional integration ($\beta \leq 0$), the estimate for α is given by:

$$\hat{\alpha}(\sigma, a) = \sup_{\beta}\{\partial_a |\mathcal{W}\{f, \psi_{-\beta}\}(\sigma, a)| = 0\}. \tag{2.21}$$

In Figure 2.6 we present the result of the monoscale analysis method on sedimentary records [13]. The top plot shows a well data set, and the second plot the singularity orders of the well data found by the monoscale analysis. The third plot shows an ice core measurement, and the bottom plot presents the singularity orders of the ice core measurement. The color defines the singularity order, and the size of the dots their masgnitude.

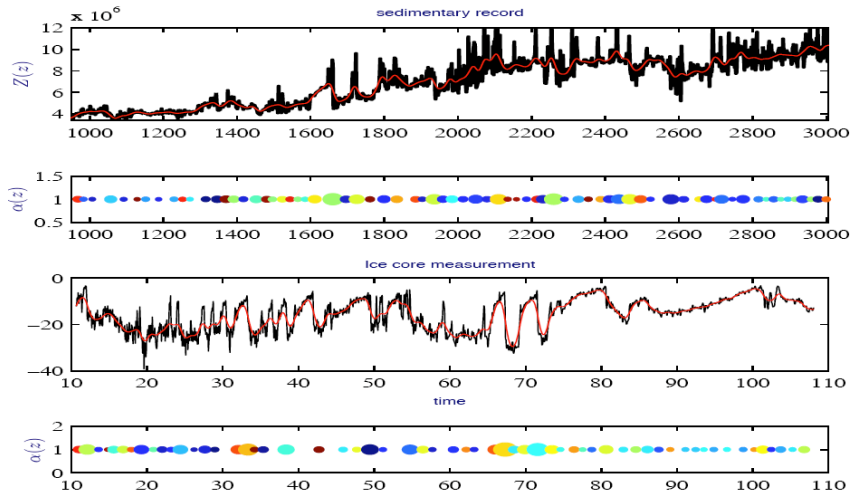

Figure 2.6: Application of the monoscale method. The top plot shows the smoothed acoustic impedance, and the second plot displays the singularity orders if the impedance found by the monoscale analysis. The third plot shows an ice core measurement, and the bottom plot displays the singularity characterization of the ice core measurement. The color defines the singularity order, and the size of the dots their masgnitude.Taken from [13]

This method gives quite good results. However, it doesn't always maintain the lateral continuity of the exponents along the reflectors, and lacks robustness and is highly sensitive to noise and any phase rotation of the waveforms in the signal.

2.2.4 Seismic Atomic decomposition

Following the monoscale analysis, Herrmann proposed in 2005 [14] a parameterization of seismic signals using redundant dictionaries. In this framework, the signal was decomposed in a dictionary of waveforms parameterized by their fractional-order, and represented by its coefficients. Assuming sparse reflectivity, the idea is to find the sparsest set of coefficients that represents the signal in the chosen dictionary. The sparsest representation contains a few large coefficients pertaining to the different seismic events in the signal. This problem was tackled using Matching Pursuit [11, 14, 21] and Basis Pursuit [2, 14], which are two algorithms that approximate the sparsest representation. The results showed the ill-posedness of the problem and the difficulties in finding the sparsest representation for complicated signals in large dictionaries. For simple configurations, both algorithms were able to find the sparsest representation, but failed when the complexity of the dictionary and the signal was too large. The definitions of dictionaries and explanations of Matching Pursuit and Basis Pursuit are given in the next chapter and Capita Selecta.

In this paper, we propose a two-step method that divides the resolution of the problem into two subproblems, that we call detection and estimation.

Chapter 3
Problem Statement

Characterizing transitions in the Earth involves finding their locations, amplitudes, and fractional exponents. In particular, given the seismic signal

$$s(z) = \sum_{i \in \Lambda_C} C_i D^{-\alpha_i} \psi(z - z_i) \pm \sum_{i \in \Lambda_A} C_{\alpha_i} D^{-\alpha_i} \psi(z - z_i)$$

as given in (2.15), the goal is to estimate the locations z_i, the amplitudes C_i and fractional exponents α_i. One of the methods proposed to solve this problem is atomic decomposition [14], whose results constitute the starting point of our detection-estimation method.

3.1 Dictionaries and decomposition

In the atomic decomposition method, the aim is to decompose the whole seismic signal uniquely with respect to a dictionary and to represent it by its coefficients. We use terminology introduced by Mallat and Zhang [21]. In $\mathbb{R}^n$, a dictionary $\mathcal{D}$ is a collection of parameterized waveforms $(\varphi_\gamma)_{\gamma \in \Gamma}$, where the waveforms φ_γ are discrete-time signals of finite length n, called *atoms*, and Γ is the set of the indexing parameter γ. In the case of a frequency or Fourier dictionary, γ is the indexing frequency, whereas for a time/scale dictionary, γ is the indexing time/scale jointly. A dictionary is called complete if it contains exactly n linearly independent atoms, or overcomplete if it is full rank and contains more than n atoms.

Given an overcomplete dictionary $\mathcal{D}$ of normalized atoms, we consider the decomposition $c =$

$(c_\gamma)_{\gamma\in\Gamma}$ of a signal s as

$$s = \sum_{\gamma\in\Gamma} c_\gamma \varphi_\gamma, \tag{3.1}$$

where the representation of the signal s is given by the vector $c = (c_\gamma)_{\gamma\in\Gamma}$. For discrete signals, we can consider the atoms of $\mathcal{D}$ as columns of a matrix $\mathbf{\Phi}$ and write equation (3.1) in the matrix form

$$\underline{\mathrm{s}} = \mathbf{\Phi}\underline{\mathrm{c}}, \tag{3.2}$$

where $\mathbf{\Phi}$ is an $n \times m$ $(m \geq n)$ matrix representing the dictionary $\mathcal{D}$, $\underline{\mathrm{s}} \in \mathbb{R}^n$ and $\underline{\mathrm{c}} \in \mathbb{R}^m$. We also use the adjoint of $\mathbf{\Phi}$ denoted by $\mathbf{\Phi}^*$. $\mathbf{\Phi}$ and $\mathbf{\Phi}^*$ satisfy the relation:

$$\forall \underline{\mathrm{x}} \in \mathbb{R}^m, \underline{\mathrm{y}} \in \mathbb{R}^n \ , \ \langle \mathbf{\Phi}\underline{\mathrm{x}}, \underline{\mathrm{y}} \rangle = \langle \underline{\mathrm{x}}, \mathbf{\Phi}^*\underline{\mathrm{y}} \rangle.$$

We adopt terminology used by Chen et al. [2]. The matrix $\mathbf{\Phi}$ represents the *synthesis* operator, which consists of building up a signal $\underline{\mathrm{s}}$ by superposing atoms of $\mathcal{D}$. The matrix $\mathbf{\Phi}^*$ represents the *analysis* operator, whose action is to find a vector of coefficients $\underline{\mathrm{c}}$ that correspond to the atoms in $\underline{\mathrm{s}}$. The normalization of the dictionary, leading to $\|\mathbf{\Phi}\| = 1$, is crucial for the convergence of any minimization involving dictionary decompositions. In the rest of the paper, we will use normalized dictionaries and will refer to $\mathbf{\Phi}$ as the matrix of $\mathcal{D}$. Finding the solution $\underline{\mathrm{c}}$ of (3.2) in an overcomplete dictionary is an underdetermined problem, and thus has an infinite number of solutions [21, 2].

3.1.1 Basis Pursuit

The nonuniqueness of $\underline{\mathrm{c}}$ enables us to choose the representation $\underline{\mathrm{c}}$ that is most suited to our problem. Even though seismic reflectivity yielded by sedimentary records is a non sparse signal as it reflects everywhere, we can treat it as a sparse signal by only considering the few large outlying reflectors. Seismic reflectivity is therefore assumed to be a sparse signal. Given sparse reflectivity, and assuming the dictionary is chosen to give a sparse representation of seismic signals, it is natural to impose sparsity on the representation $\underline{\mathrm{c}}$ and choose the sparsest $\underline{\mathrm{c}}$ that represents $\underline{\mathrm{s}}$. We define an event to be any localized waveform in $\mathbb{R}^n$.

The sparsest representation is the representation with minimum ℓ^0 norm (i.e. the fewest number of non-zero entries), which is referred to as the solution of the (P_0) problem [7]:

$$(P_0) \ \min_{\underline{c}} \|\underline{c}\|_0 \text{ subject to } \underline{s} = \mathbf{\Phi}\underline{c}. \tag{3.3}$$

For the remainder of this paper, we will describe the sparsest solution as the solution of the (P_0) problem. Solving the (P_0) problem is NP-hard, because its computational complexity increases exponentially with the number of atoms in $\mathcal{D}$ [2]. Basis Pursuit (BP) [2, 7] approximates the sparsest representation by convexifying the (P_0) problem into the following problem:

$$(P_1) \ \min_{\underline{c}} \|\underline{c}\|_1 \text{ subject to } \underline{s} = \mathbf{\Phi}\underline{c}. \tag{3.4}$$

For seismic signals, the decomposition involves a dictionary of waveforms parameterized by their fractional-order α, as opposed to a single wavelet in spiky decon*. Examples of atoms chosen for seismic applications are fractional B splines and fractional spline wavelets [14]. The idea is to find the few atoms in $\mathcal{D}$ that best match the different events in $\underline{s}$, and deduce the fractional-order of the events from these atoms. Herrmann [14] proposed an atomic decomposition of seismic signals with BP, and showed that BP manages to find the sparsest representation when the dictionary is small or the signal simple. However, as shown by Herrmann [14], the larger the dictionary, the more similar the atoms in $\mathcal{D}$, so if the dictionary gets too large, BP will have difficulties distinguishing between similar waveforms and hence not find the sparsest solution.

3.2 Uniqueness condition and equivalence between (P_0) and (P_1)

Herrmann [14] showed empirically that for certain configurations of signals and dictionaries, the solution to the (P_1) problem was the solution to the (P_0) problem, which, in general, is not the case. There exist some conditions [7] under which a representation $\underline{c}$ is the unique sparsest representation, and stronger conditions under which the (P_0) and (P_1) problems are equivalent. To discuss these conditions, we need to introduce the following definitions.

Definition 3.2.1 (Spark [7]) *Let* $\mathbf{\Phi}$ *be a matrix of size* n *by* m*, with* $m \geq n$*. The Spark of* $\mathbf{\Phi}$ *is defined as the smallest possible number* σ*, such that there exists a subgroup of* σ *columns from* $\mathbf{\Phi}$ *that are linearly dependent.*

Definition 3.2.2 ($\mu_{1/2}(\mathbf{G})$ [7]) *For* $\mathbf{G}$ *a symmetric matrix,* $\mu_{1/2}(\mathbf{G})$ *denotes the smallest integer* k *such that some collection of* k *off-diagonal magnitudes arising in a single row or column of* $\mathbf{G}$ *sums at least to* $\frac{1}{2}$*.*

Although Spark and rank of a matrix are in some ways similar [7], they remain two separate concepts:

Example 3.2.1 *We show a full rank matrix* $\mathbf{B}$ *with a Spark 2.*

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Clearly $\mathbf{B}$ *is full rank as it has 4 independent columns (*$rank(\mathbf{B}) = 4$*), but* $\mathbf{B}$ *also has two identical columns which reduces its Spark to 2.*

The Spark is an important concept that measures the similarity between any two columns of $\mathbf{\Phi}$, which for dictionaries, translates to the similarity between any two atoms in $\mathcal{D}$. The uniqueness condition involves the Spark of $\mathcal{D}$, and is given in the following theorem:

Theorem 3.2.1 (Uniqueness [7]) *Consider a dictionary* $\mathcal{D}$ *with matrix* $\mathbf{\Phi}$ *If there exists a representation* $\underline{c}_0$ *of* $\underline{s}$ *, such that* $\underline{s} = \mathbf{\Phi}\underline{c}_0$ *and* $\|\underline{c}_0\|_0 < \sigma/2$ *, then* $\underline{c}_0$ *is the sparsest solution possible, i.e.* $\underline{c}_0$ *solves* (P_0) *problem.*

Given a signal s and dictionary $\mathcal{D}$, Theorem 3.2.1 states the sparsity condition under which a representation $\underline{c}$ is the unique sparsest representation of s in $\mathcal{D}$. If a representation $\underline{c}$ has

strictly less than $\sigma/2$ non-zero entries, it is the sparsest representation. However, this theorem doesn't show whether this sparsest representation can be the solution to an ℓ^1 minimization, which is stated in Theorem 3.2.2.

The equivalence between the (P_0) and (P_1) problems involves the quantity $\mu_{1/2}(\mathbf{G})$ ($\mu_{1/2}(\mathbf{G}) < \sigma$), where $\mathbf{G}$ is the Gram matrix of $\mathbf{\Phi}$ defined by $\mathbf{G} = \mathbf{\Phi}^*\mathbf{\Phi}$. $\mathbf{\Phi}^*$ denotes the adjoint of $\mathbf{\Phi}$. The following theorem gives the equivalence condition [7]:

Theorem 3.2.2 *[7] Consider a dictionary $\mathcal{D}$ with matrix $\mathbf{\Phi}$ and Gram matrix G. If there exists a representation $\underline{c}_0$ such that $\underline{s} = \mathbf{\Phi}\underline{c}_0$ and $\|\underline{c}_0\|_0 < \mu_{1/2}(\mathbf{G})$, then $\underline{c}_0$ is the unique solution of (P_1) and is the sparsest solution.*

For a given signal $\underline{s}$ and dictionary $\mathcal{D}$, solving the (P_1) problem gives the unique sparsest solution, only if its solution is sparse enough in $\mathcal{D}$ (i.e. its ℓ^0 norm is strictly less than the quantity $\mu_{1/2}(\mathbf{G})$). The choice of $\mathcal{D}$ is hence very important as it determines the sparsity condition under which the solution of (P_1) is also solution of (P_0). However, the condition stated in theorem 3.2.2 is necessary, but not sufficient, as solutions to (P_1) with ℓ^0 norm larger than $\mu_{1/2}(\mathbf{G})$) may still be solutions to (P_0). When solutions to (P_1) do not satisfy the condition of theorem 3.2.2, the situation is less clear, as such solutions may or may not be solutions to (P_0). In our case and for the type of dictionaries we use, solutions to (P_1) often do not satisfy the condition of theorem 3.2.2. However, Starck et al. [25] empirically demonstrated the gap between theoretical results (based on worst-case analysis), and the empirical evidence of BP's performance. The goal of our new detection-estimation method is to increase the empirical bounds shown for Basis Pursuit

3.2.1 Empirical studies for the performance of BP

In many theoretical and practical applications, it is necessary to know the value of the Spark and $\mu_{1/2}(\mathbf{G})$ for a given dictionary $\mathcal{D}$. However, computing these quantities involves several ℓ^0 minimizations, and requires enumerating subsets of columns of $\mathbf{\Phi}$, which is computationally

intractable as the algorithm complexity grows exponentially with the size of $\mathcal{D}$ [7]. Donoho et al. [7] proposed a numerical scheme for computing an upper bound on the Spark, by replacing the ℓ^0 minimizations by ℓ^1 minimizations. However, although the scheme is now computationally tractable, its numerical results are very difficult to interpret [7]. Due to this computational barrier, the performance of BP must be evaluated through numerical experiments.

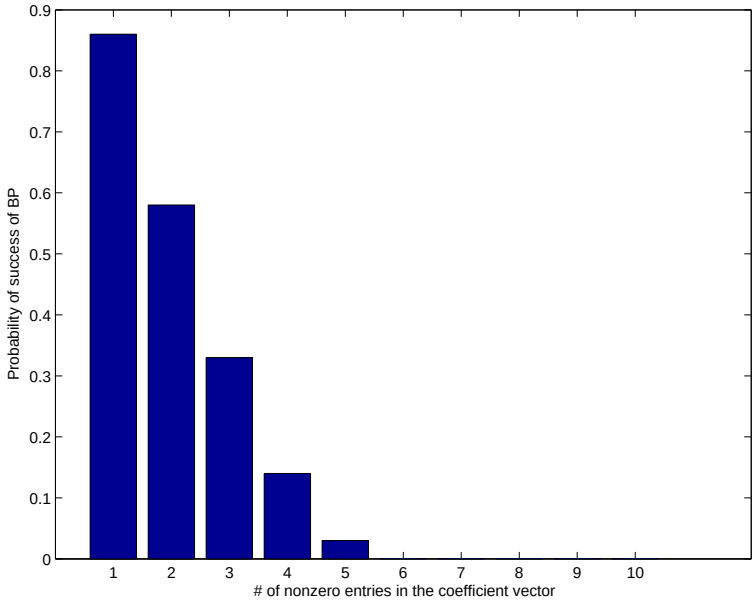

Figure 3.1: Probability of success of BP for different number of non-zero entries in the representation $\underline{c}$. We used a fractional spline wavelet dictionary with ten different fractional-orders α ranging from 0.5 to 2. One can clearly see that the probability of success of BP decreases very fast with the number of non-zero entries in the coefficient vector.

We performed simulations with fractional spline wavelet dictionaries used to decompose seismic signals [14], and showed that, as soon as the representation $\underline{c}$ had more than two non-zero entries, the probability of success of BP was very low. This is illustrated in Figure 3.1. These results explain the failure of Herrmann's atomic decomposition [14] for large dictionaries and complicated signals, and show that for large *seismic dictionaries*, the representation of the signal must be highly sparse (its representation $\underline{c}$ must satisfy $\|\underline{c}\|_0 < 2$) in order to be the unique solution of an ℓ^1 minimization.

3.3 Detection-estimation approach

We address the above issues through a two-step approach: we first partition the seismic signal $\underline{s}$ into its major features, and then analyze each of them separately by performing their ℓ^1 decomposition in a large dictionary of waveforms parameterized by their fractional-order. Each major feature, containing only a few events, is now a *simple* and localized signal that can be successfully decomposed with respect to large dictionaries. We consider dictionaries made with all translations of parameterized waveforms. For example, a dictionary containing waveforms parameterized by two different fractional-orders α_1 and α_2 will be written as

$$\mathcal{D} = [\mathcal{D}_{\alpha_1}, \mathcal{D}_{\alpha 2}], \tag{3.5}$$

where $\mathcal{D}_{\alpha_1}$ contains all the translations of a waveform parameterized by α_1, and $\mathcal{D}_{\alpha_2}$ contains all the translations of a waveform parameterized by α_2. In a matrix form, equation (3.5) can be written as

$$\mathbf{\Phi} = [\mathbf{\Phi}_{\boldsymbol{\alpha_1}}, \mathbf{\Phi}_{\boldsymbol{\alpha_2}}]. \tag{3.6}$$

Due to this construction, the synthesis operator associated to $\mathbf{\Phi}$ acts as a convolution, and the analysis operator associated to $\mathbf{\Phi}^*$ as a correlation.

Chapter 4
Detection-estimation Method

We now discuss our approach to the problem of delineating and characterizing transitions from seismic data. Studies on sedimentary records have shown that seismic reflectivity is made of accumulation of varying order singularities, leading to a non sparse signal. However, unlike white noise, sedimentary records clearly contain distinct large outlying reflectors with strong singularities evidienced by small fractional-orders, giving rise to a few large "*spiky*" events in the reflectivity. In this paper, we are only interested in finding these outlying reflectors, and consider the reflectivity as a function of these large outliers. We therefore assume that the reflectivity contains a few events and denote the seismic signal $\underline{s}$.

The principle of our method is to divide the original problem into two easier subproblems. We start by partitioning the signal $\underline{s}$ into its major features (pertaining to the predominant reflectors in the Earth), and then proceed by analyzing and characterizing each feature separately. We name these two steps detection and estimation respectively.

4.1 Detection

The purpose of the detection is to find the locations and characteristic scales of the major features in the signal $\underline{s}$. In particular, the detection step answers two specific questions:

1. What is the partition of the signal $\underline{s}$ into its prevailing components?

2. What is the characteristic scale of each detected component?

This information is then used to build windowing functions to extract the major components from the signal $\underline{s}$. The principle of the detection is to find an approximate sparse representation of the seismic signal $\underline{s}$ in an appropriate dictionary, and select the largest sets of coefficients that correspond to the predominant features in $\underline{s}$.

4.1.1 Detection dictionary

Like any matched-filter approach, it is important to choose atoms that best correlate with the events in $\underline{s}$ in order to guarantee the sparsity of its representation. To satisfy this condition, we choose to estimate the average waveform of $\underline{s}$, by finding the waveform φ_α whose Fourier spectrum best fits the Fourier spectrum of the signal $\underline{s}$. Because we work with waveforms parameterized by their fractional-order (and implicitly or explicitly parameterized by their frequency), we estimate the average waveform in terms of its fractional-order $\bar{\alpha}$, and central frequency $\bar{f}$. We construct a larger detection dictionary $\mathcal{D}_d$ by selecting the estimated average waveform $\varphi_{\bar{\alpha}}$ and some *neighboring* ones. By *neighboring* waveforms, we refer to waveforms whose parameter α and frequency f are close to the parameter $\bar{\alpha}$ and frequency $\bar{f}$ of the average waveform. We choose α and f such that

$$|\alpha - \bar{\alpha}| \leq \eta \qquad \text{and} \qquad \epsilon_1 \leq \frac{f}{\bar{f}} \leq \epsilon_2.$$

From experiments we choose $\eta = 1$, $\epsilon_1 = \frac{1}{2}$ and $\epsilon_2 = 2$. The choices we made for η, ϵ_1 and ϵ_2 are not fixed, and can be modified, as long as the *neighboring* waveforms obtained from these changes still correlate well with the seismic signal. Adding more fractional-orders and more frequency content in the detection dictionary, accounts for the non-stationary properties of the seismic signal, and ensures the sparsity of its representation with respect to $\mathcal{D}_d$ (multiple spiky deconvolutions). The detection dictionary $\mathcal{D}_d$ is therefore made of the selected waveforms (average waveforms and *neighboring* ones) and all their translations, and can be seen as the concatenation of several dictionaries.

For clarification, consider a detection dictionary $\mathcal{D}_d$ made of p selected waveforms and their

translations. $\mathcal{D}_d$ can be written as

$$\mathcal{D}_d = [\mathcal{D}_{d_1}, ..., \mathcal{D}_{d_p}], \tag{4.1}$$

where $\mathcal{D}_{d_i}$ is the detection dictionary made of the ith waveform and all its translations. For discrete signals, $\mathcal{D}_d$ can be represented by its matrix $\mathbf{\Phi_d}$, which can also be written as a concatenation of p matrices

$$\mathbf{\Phi_d} = [\mathbf{\Phi_{d_1}}, ..., \mathbf{\Phi_{d_p}}], \tag{4.2}$$

where $\mathbf{\Phi_{d_i}}$ is the matrix of $\mathcal{D}_{d_i}$.

4.1.2 Decomposition in detection

We envision an approximate sparse decomposition

$$s = \sum_{i=1}^{M} c_{\alpha_i} \varphi_{\alpha_i} + r^{(M)}, \tag{4.3}$$

where $r^{(M)}$ is the residual and $c = (c_{\alpha_i})_{i=1}^{M}$ the representation of s with respect to the detection dictionary. This approximation selects the M largest coefficients c_{α_i}, which correspond to the M main features of the signal s, and approximately represent s in a sparse solution c. Because the aim of the detection step is to extract only the major features of the signal s, it is sufficient to approximately reconstruct the signal s and consider the above approximate decomposition. Equation (4.3) can be written in a matrix form as

$$\underline{\mathrm{s}} = \mathbf{\Phi_d}\underline{\mathrm{c}} + \underline{\mathrm{r}}, \tag{4.4}$$

where $\mathbf{\Phi_d}$ is the matrix of $\mathcal{D}_d$ of size $n \times m$ $(m \geq n)$, $\underline{\mathrm{s}}$ and $\underline{\mathrm{r}} \in \mathbb{R}^n$ and $\underline{\mathrm{c}} \in \mathbb{R}^m$.

$\underline{\mathrm{c}}$ refers to the solution of

$$\min_{\underline{\mathrm{c}}} \frac{1}{2} \|\underline{\mathrm{s}} - \mathbf{\Phi_d}\underline{\mathrm{c}}\|_2^2 + \lambda_d \|\underline{\mathrm{c}}\|_1, \tag{4.5}$$

where $\lambda_d > 0$ is a trade-off parameter between the ℓ^1 norm of $\underline{\mathrm{c}}$ and the data misfit, $\mathbf{\Phi_d} = [\mathbf{\Phi_{d_1}}, ..., \mathbf{\Phi_{d_p}}]$, and $\underline{\mathrm{c}} = [\underline{\mathrm{c}}_1, ..., \underline{\mathrm{c}}_p]^T$. The size of the residual and hence the number of selected

atoms M in the representation c, is controlled by λ_d. The residual goes to zero as $\lambda_d \rightarrow 0$, leading to an exact reconstruction. On the other hand, if $\lambda_d \rightarrow \infty$, the residual gets large, with $\lim_{\lambda_d \rightarrow +\infty} \underline{\mathrm{r}} = \underline{\mathrm{s}}$, and $\lim_{\lambda_d \rightarrow +\infty} \underline{\mathrm{c}} = 0$. For noisy data, (4.5) becomes a denoising problem, where the value of λ_d has to be set according to the noise level. Chen et al. [2] proposed to set

$$\lambda_d = \sigma_n \sqrt{2log(p)}, \tag{4.6}$$

where σ_n is the noise level, and p the cardinality of the dictionary. Chen et al. [2] refer to (4.5) as Basis Pursuit De-Noising (BPDN) [2]. Similarly to BP, BPDN can be rewritten as a perturbed Linear Programming problem, and is solved using the Simplex algorithm or an Interior-Point scheme [2].

4.1.3 Algorithm

Because of limitations associated with the Basis Pursuit and Matching Pursuit methods that we will discuss later, we propose to solve (4.5) using an iterative thresholding algorithm [4] on each coefficient $\underline{\mathrm{c}}_i$. The resulting algorithm amounts to a Block Coordinate Relaxation Method [24, 25] performed on $\underline{\mathrm{c}}$, with a Landweber iteration with thresholding applied at each iteration step on each $\underline{\mathrm{c}}_i$ [4]. At each iteration k we minimize the functional

$$\frac{1}{2}\|\underline{\mathrm{s}} - \boldsymbol{\Phi_d}\underline{\mathrm{c}}\|_2^2 + \lambda_k \|\underline{\mathrm{c}}\|_1, \tag{4.7}$$

with respect to the coefficient $\underline{\mathrm{c}}$, where $\lambda_k > 0$ is a decreasing parameter. We start with a large λ_k and then decrease it to some predetermined minimum value λ_{min}. The choice of λ_{min} depends on how small we want the data misfit to be and is specified at the beginning of the algorithm. A small λ_{min} will give a small data misfit and an exact reconstruction, whereas a large λ_{min} will contribute to a large data misfit and an approximate reconstruction. For normalized signals, and to a good approximation, we choose $\lambda_{min} = 10^{-4}$.

At each iteration k, we solve for the vector $\underline{\mathrm{c}}^k = [\underline{\mathrm{c}}_1^k, ..., \underline{\mathrm{c}}_p^k]$, by separately solving for its different components $\underline{\mathrm{c}}_i^k$, and assuming the other components $\underline{\mathrm{c}}_j^k$, $j \neq i$ are fixed. The representation $\underline{\mathrm{c}}_i^k$

is updated according to the rule [4, 25]

$$\underline{c}_i^{k+1} = \mathcal{S}_{\lambda_k}(c_i^k + \boldsymbol{\Phi_{d_i}}(\underline{R}_i^k - \boldsymbol{\Phi_{d_i}}^* \underline{c}_i^k)), \tag{4.8}$$

where $\boldsymbol{\Phi_{d_i}}^*$ denotes the adjoint operator of $\boldsymbol{\Phi_{d_i}}$ and $\underline{R}_i^k = \underline{s} - \sum_{j \neq i} \boldsymbol{\Phi_{d_j}}^* \underline{c}_j^k$. $\mathcal{S}_{\lambda_k}$ is the componentwise soft thresholding operator by λ_k, defined by $\mathcal{S}_{\lambda_k}(x) = sign(x)(|x| - \lambda_k)_+$, for $x \in \mathbb{R}$, where $y_+ = max(0, y)$ for $y \in \mathbb{R}$.

The choice of the iterative thresholding algorithm to solve (4.5) is motivated by the algorithm's flexibility and control over the decrease of the parameter λ_k and choice of λ_{min}. The way the parameter λ_k decreases determines the accuracy of the algorithm: the slower the decrease, the more accurate the solution $\underline{c}$. In our case, we are only interested in the major features in $\underline{s}$, which correspond to the larger coefficients in $\underline{c}$. For that reason, we choose to use an exponential decrease of the parameter λ_k which will make the algorithm slowly capture the large coefficients at the beginning of the iterations, and then move on faster on the small coefficients. The expression for the exponential decrease of λ_k is given by:

$$\lambda_k = A - \mathrm{e}^{-ks}, \tag{4.9}$$

where $A = 1 + \lambda_0$ and s is a predetermined speed parameter.

The representation $\underline{c}$, obtained by iteratively minimizing (4.7), is numerically not sparse, as none of its entries is strictly zero. However, most of them are within a tolerance of $10^{-6} \ll \lambda_{min}$, which is negligible. We now present our approach to detect the major events in $\underline{s}$.

According to the notations introduced in equation (4.2), the representation $\underline{c}$ obtained by minimizing (4.5) is a vector of length m ($m = pn$) that can be written as

$$\underline{c} = [\underline{c}_1, ..., \underline{c}_p],\ \underline{c}_i \in \mathcal{D}_i,$$

where each $\underline{c}_i \in \mathbb{R}^n$. In order to incorporate the contribution of each dictionary $\mathcal{D}_i$ in the detection process, we consider the different vectors $\underline{c}_i$ as columns of a matrix $\boldsymbol{C}$. Each row k in $\boldsymbol{C}$, corresponding the location k in $\underline{s}$, contains the coefficients of $\underline{s}$ with respect to the different

dictionaries $\mathcal{D}_i$ at the location k. By taking the largest entry in the magnitude of each row, we construct another vector $\underline{c}_T$

$$\forall\ 1 \leq k \leq n\ ,\ \underline{c}_T(k) = \max_{1 \leq i \leq p} |\boldsymbol{C}(k,i)| = \max_{1 \leq i \leq p} |\underline{c}_i(k)|,$$

where $\underline{c}_T \in \mathbb{R}^n$ and $\boldsymbol{C} \in n \times p$. $\underline{c}_T$ decries the best contribution of the dictionaries $\mathcal{D}_i$ at the different locations in $\underline{s}$, and the largest entries in $\underline{c}_T$ correspond to the major features in $\underline{s}$.

We now select the largest entries of $\underline{c}_T$, by only considering those larger than a threshold μ. The choice of μ depends on the type of signals involved, and is very important, especially when dealing with noisy data. μ is chosen to be a percentage of the ℓ^2 norm of $\underline{c}_T$, written as

$$\mu = \rho \|\underline{c}_T\|_2,$$

where ρ is the percentage. From experimental results, we choose to set $\rho = 0.1$, but this value can be modified. For noisy data for example, ρ should be set at a large enough value in order to be above the noise level. Note that for noisy data, λ_{min} can also be set to a larger value depending on the noise level, as stated in (4.6). We call $\underline{c}_T^{t_\mu}$ the thresholded vector $\underline{c}_T$ with threshold level μ. For clarity purposes, we omit the index T and denote the thresholded vector $\underline{c}_T^{t_\mu}$ as $\underline{c}_{t_\mu}$.

4.1.4 Windowing

The next step in the detection is the analysis of the thresholded vector $\underline{c}_{t_\mu}$, and the selection of the major features in $\underline{s}$. The representation $\underline{c}_{t_\mu}$ contains localized sets of nonzero entries corresponding to the main components of $\underline{s}$, and can be written as

$$\underline{c}_{t\mu} = \sum_{i=1}^{M} \underline{c}_{t_\mu}^i, \tag{4.10}$$

where M is the number of sets in $\underline{c}_{t\mu}$, $\underline{c}_{t_\mu}^i$ is a vector of length n containing the ith set of largest nonzero entries in $\underline{c}_{t\mu}$, and $\forall i \neq j$, $\sup(\underline{c}_{t_\mu}^i) \cap \sup(\underline{c}_{t_\mu}^j) = \emptyset$.

An example of such a partition is given below:

$$\underline{c}_{t_\mu} = (0, 1, 2, 0, 0, -1, -2, 0) \tag{4.11a}$$

$$\underline{c}^1_{t_\mu} = (0, 1, 2, 0, 0, 0, 0, 0) \tag{4.11b}$$

$$\underline{c}^2_{t_\mu} = (0, 0, 0, 0, 0, -1, -2, 0) \tag{4.11c}$$

We now define m_i as the largest entry in the modulus of $\underline{c}^i_{t_\mu}$

$$m_i = \max_l |\underline{c}^i_{t_\mu}(l)| \tag{4.12}$$

and denote by φ_{α_i} the atom in $\mathcal{D}$ that corresponds to the entry m_i. Each vector $\underline{c}^i_{t_\mu}$ describes a particular feature in $\underline{s}$ which does not overlap with the other ones, as stated by the condition $\forall i \neq j$, $\sup(\underline{c}^i_{t_\mu}) \cap \sup(\underline{c}^j_{t_\mu}) = \emptyset$. The location l_i and frequency f_i of each atom φ_{α_i} determine the location and characteristic scale of the ith main feature in $\underline{s}$.

We now discuss the problem of extracting these detected features from the signal $\underline{s}$. We adopt the methodology of multiplying $\underline{c}_{t_\mu}$ by localized windows centered at the locations l_i. We introduce the following notations.

Let $\mathcal{W}_i$ be the window associated to the ith selected atom φ_{α_i}, centered at l_i. Each window $\mathcal{W}_i$ is chosen to be a discrete function given by

$$\mathcal{W}_i(k) = \begin{cases} 0 & \text{if } 0 \leq k < l_i - (\beta + \delta)\sigma_i \\ exp(\frac{-(n-(l_i-\beta\sigma_i-1))^2}{2\sigma^2_{g_i}}) & \text{if } \beta\sigma_i < l_i - k \leq (\beta + \delta)\sigma_i \\ 1 & \text{if } |k - l_i| \leq \beta\sigma_i \\ exp(\frac{-(n-(l_i+\beta\sigma_i+1))^2}{2\sigma^2_{g_i}}) & \text{if } \beta\sigma_i < k - l_i \leq (\beta + \delta)\sigma_i \\ 0 & \text{if } l_i + (\beta + \delta)\sigma_i < k \leq n, \end{cases} \tag{4.13}$$

where σ_i is the standard deviation of $\varphi_{\alpha i}$, and σ_{g_i} is the standard deviation of the Gaussian function used for the decay of the window at the tips.

The parameters β and δ specify the width of the window. We provide the illustration of the window in figure 4.1. We choose the default values to be $\beta = 2$ and $\delta = 2$, as $4\sigma_i$ is a good

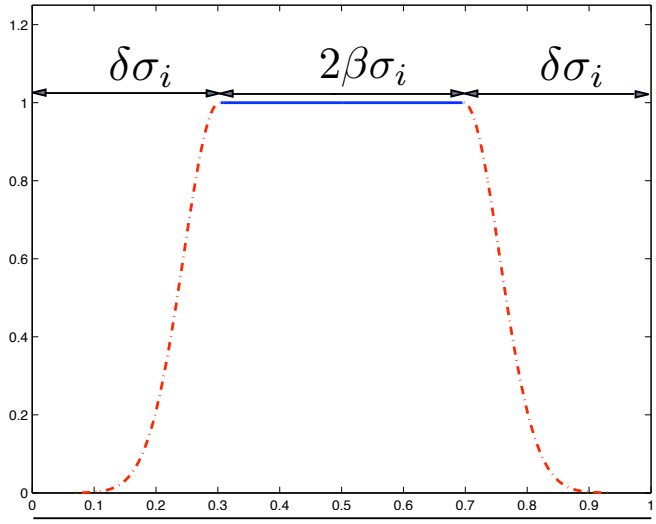

Figure 4.1: The ith window and its associated parameters β and δ. The function plotted is multiplited to the signal to extract the ith detected signal component. The parameters β and δ are as given in equation (4.13)

approximation for the width of φ_{α_i}. The window is equal to 1 over the width $2\beta\sigma_i$ centered at l_i. $\delta\sigma_i = 2\sigma_i$ corresponds to the width of the Gaussian decay at both tips of the window, which is a choice motivated by numerical experiments. The window ends at both tips with a Gaussian decay with a width equal to $\delta\sigma_i$ for each decay. Both β and δ depend on the data and should be modified accordingly. So far, the choices we made have given good results.

By multiplying the signal $\underline{s}$ by each window $\mathcal{W}_i$, we partition $\underline{s}$ into its prevailing components $\underline{s}_i$, where $\underline{s}_i \in \mathbb{R}^n$. The multiplication between the signal $\underline{s}$ and the window $\mathcal{W}_i$ is done componentwise, such that

$$\forall\ 1 \leq k \leq n\ ,\ \underline{s}_i(k) = \underline{s}(k)\mathcal{W}_i(k). \tag{4.14}$$

Each signal $\underline{s}_i$ is now a simple and localized signal assumed to be a superposition of a small number of atoms. We show an example of a windowing in Figure 4.2. The top plot shows the signal and the window applied to one detected event. The bottom plot shows the windowed event.

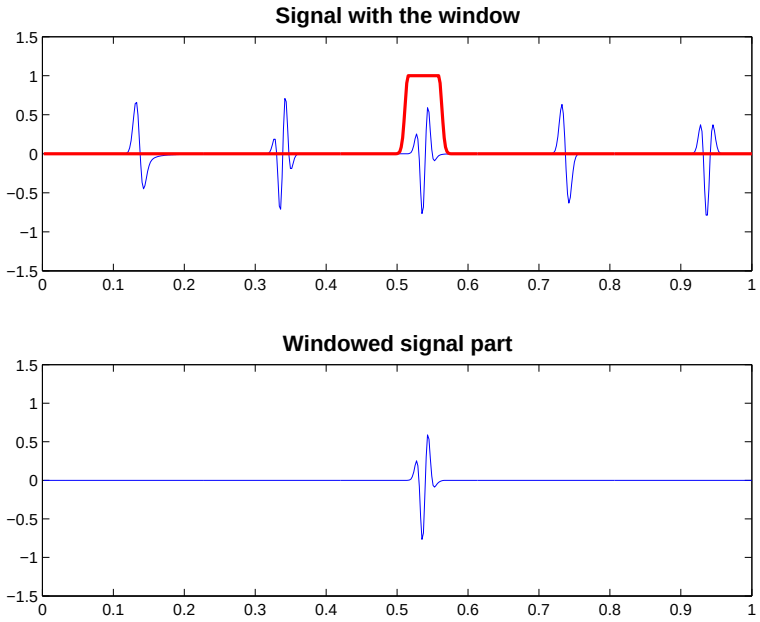

Figure 4.2: The top plot shows the signal and the window applied to one detected event. The bottom plot shows the windowed event.

4.2 Estimation

We now treat each windowed signal s_i as a generic signal s. The estimation algorithm operates with simple and localized signals s assumed to be a superposition of a small number of waveforms parameterized by their fractional-order. The purpose of the estimation is to unravel and approximate the fractional-order of these waveforms, by decomposing the localized signal with respect to a large dictionary. In particular, the estimation step answers two quantitative questions:

1. How many waveforms does s contain?

2. What is the fractional-order of each of these waveforms?

4.2.1 Decomposition in estimation

We consider a signal s as

$$s = \sum_{i=1}^{K} a_{\alpha_i} \phi_{\alpha_i}, \tag{4.15}$$

where ϕ_{α_i} is a localized waveform parameterized by its fractional-order α_i, and K a small integer, leading to a very sparse representation $\underline{a} = (a_{\alpha_i})_{i=1}^{K}$. We denote $\mathcal{D}_K$ the dictionary containing the waveforms $(\phi_{\alpha_i})_{i=1}^{K}$. The principle of the estimation step is to find the K waveforms that make up the signal $\underline{s}$ (or an approximation of these waveforms), through the decomposition of s in a large estimation dictionary of waveforms parameterized by their fractional-order.

We envision the decomposition of the signal s as

$$s = \sum_{i=1}^{N} c_{\alpha_i} \varphi_{\alpha_i}, \tag{4.16}$$

where $(\varphi_{\alpha_i})_{i=1}^{N}$ are waveforms taken from an overcomplete estimation dictionary $\mathcal{D}_e$, and $c = (c_{\alpha_i})_{i=1}^{N}$ is a very sparse representation of s. Equation (4.16) can be written in a matrix form as

$$\underline{s} = \mathbf{\Phi_e} \underline{c}, \tag{4.17}$$

where $\mathbf{\Phi_e}$ is the matrix of $\mathcal{D}_e$ of size $n \times m$ $(m \geq n)$, $\underline{s} \in \mathbb{R}^n$ and $\underline{c} \in \mathbb{R}^m$. In the ideal case, where the estimation dictionary $\mathcal{D}_e$ contains the waveforms $(\phi_{\alpha_i})_{i=1}^{K}$, (i.e. $\mathcal{D}_K \subseteq \mathcal{D}_e$), (4.15) becomes (4.16), and the representation $\underline{a}$ coincides with the representation $\underline{c}$. When $\mathcal{D}_e$ contains a subset $\mathcal{D}_P$ of $\mathcal{D}_K$, $\mathcal{D}_P = (\phi_{\alpha_{\sigma_j}})_{j=1}^{P}$, $P < K$ and $\sigma_j \in \{1, ..., K\}$, both representations $\underline{a}$ and $\underline{c}$ coincide on the subset of coefficients that correspond to the waveforms $(\phi_\alpha)_{\alpha \in S}$. Because $\mathcal{D}_K$ is unknown, $\mathcal{D}_e$ does not in general contain the waveforms $(\phi_{\alpha_i})_{i=1}^{K}$. However, if a subset of waveforms in $\mathcal{D}_e$ converges to the waveforms in $\mathcal{D}_K$, we hope to have the representation $\underline{c}$ converge to $\underline{a}$. As stated in section 3.1, there is no unique solution $\underline{c}$ to the underdetermined problem (4.17), and we cannot guarantee that the representation $\underline{c}$ will converge to $\underline{a}$. However, following the sparsity of $\underline{a}$, and by imposing sparsity on the representation $\underline{c}$, we hope to find the representation $\underline{c}$ that will converge to $\underline{a}$ in the case where a subset of waveforms in $\mathcal{D}_e$ converges to the waveforms in $\mathcal{D}_K$. The sparsity assumption of $\underline{c}$ naturally leads to solving the (P_0) problem. However, the (P_0) problem is in genereal computationally intractable [2, 7], so we consider the easier (P_1) problem:

$$\min_{\underline{c}} \|\underline{c}\|_1 \text{ subject to } \underline{s} = \mathbf{\Phi_e} \underline{c}, \tag{4.18}$$

The result from theorem 3.2.2 given in section 3.2, describes the sparsity condition under which the solution to (P_0) is also solution to (P_1):

If there exists a representation $\underline{c}_0$ *such that* $\underline{s} = \mathbf{\Phi}\underline{c}_0$ *and* $\|\underline{c}_0\|_0 < \mu_{1/2}(\mathbf{G})$*, then* $\underline{c}_0$ *is the unique solution of* (P_1) *and is the sparsest solution.*

For large dictionaries, the above condition requires a high sparsity of the solution of (P_0), in order to guarantee its being solution to (P_1) as well. Theorem 3.2.2 reveals the trade-off between the size of the dictionary and the complexity of the signal. The larger the dictionary, the smaller $\mu_{1/2}(\mathbf{G})$, and the sparser $\underline{c}_0$ needs to be in order to also be solution of (P_1). On the other hand, $\mu_{1/2}(\mathbf{G})$ increases as the size of the dictionary decreases, which relaxes the sparsity of $\underline{c}_0$. The choice of the dictionary $\mathcal{D}_e$ is thus very important.

4.3 Dictionary in estimation

We now discuss our choice of dictionary $\mathcal{D}$. Based on the strong sparsity of $\underline{c}$ assumed in equation (4.17), we can relax the size of the dictionary and allow for a large one. Following the argument developed in section 4.1.1, it is necessary to select atoms that best correlate with the signal $\underline{s}$. In particular, we choose atoms that are waveforms parameterized by their fractional-order α, leading to

$$\mathcal{D} = (\varphi_\alpha)_{\alpha \in \mathcal{A}}, \tag{4.19}$$

where $\mathcal{A}$ is the indexing set of fractional-orders taken from the singularity spectrum $f(\alpha)$ given by a multiscale analysis [10, 20] on the seismic signal. In the estimation, it is crucial to obtain an accurate estimate of the fractional-orders contained in the signal s we analyze. For that purpose, it is necessary that the estimation dictionary contains a large number of fractional-orders α sufficiently close. There is a trade-off between the size of the dictionary and the accuracy of the estimation algorithm. When the fractional-orders in the dictionary are too far apart, we lose accuracy. On the other hand, if we choose the fractional-orders of the dictionary too close together, the dictionary becomes too large and the algorithm prohibitively slow. For our purposes, we choose the distance between 2 consecutive fractional-orders in the dictionary

to be 0.2, as it gives a good compromise between accuracy and speed. Figure 4.3 shows two waveforms with fractional-orders differing by 0.2. The waveforms are fractional derivatives of the Gaussian, whose fractional-orders are the fractional amounts of differentiation performed on the Gaussian. The top plot shows a waveform with fractional-order 2.2 and the bottom plot shows a waveform with fractional-order 2.4. Visually, these waveforms look similar, but our algorithm can still distinguish between them. Waveforms with closer fractional-orders will not be distinguished by the algorith, but this is sufficient for our purposes.

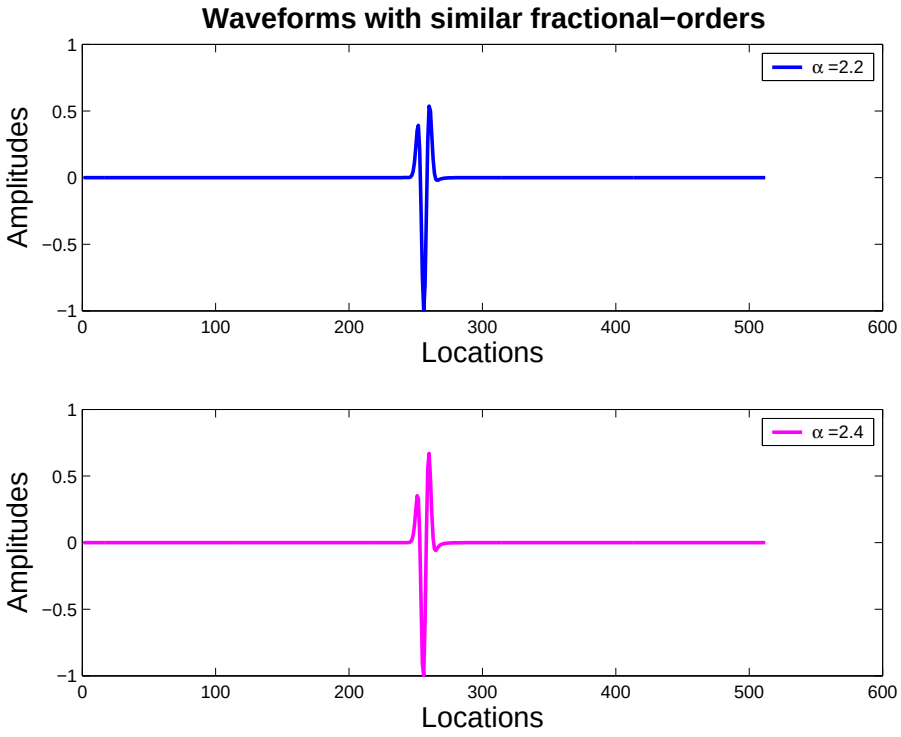

Figure 4.3: Waveforms with similar fractional-orders. Here we see almost two identical waveforms with 2 different fractional-orders differing by 0.2. The top plot shows a waveform with fractional-order 2.2, and the bottom plot displays a waveform with fractional-order 2.4. We choose the estimation dictionary so that our algorithm can distinguish 2 waveforms with fractional-order differing at least by 0.2. If the fractional-orders are closer than 0.2, our method will not distinguish them.

The same argument applies to dictionaries explicitly parameterized by fractional-orders and frequencies.

4.3.1 Resolution

We propose to solve (4.18) using the iterative thresholding algorithm [4] described in section 4.1.3. Recall the iterative thresholding algorithm, whose solution $\hat{\underline{c}}$ is given by

$$\hat{\underline{c}} = \arg\min_{\underline{c}} \big(\frac{1}{2}\|\underline{s} - \boldsymbol{\Phi_e}\underline{c}\|_2^2 + \lambda_e\|\underline{c}\|_1\big), \tag{4.20}$$

where $\underline{s} = \boldsymbol{\Phi_e}\underline{c} + \underline{r}$. In the estimation step, the signal s is decomposed with respect to a large dictionary. Unlike the detection, where we considered an approximate decomposition, the signal s is entirely decomposed, leading to an exact decomposition of s. This is very important as we want to estimate the fractional-orders of $\underline{s}$. We can solve (4.18) using the iterative thresholding algorithm, because, when $\lambda_e \to 0$, the residual $\underline{r}$ goes to zero and the solution $\hat{\underline{c}}$ behaves exactly like BP applied to $\underline{s}$. From experiments, we choose $\lambda_{min} = 10^{-10}$.

Although numerically non sparse, most of the coefficients of the representation $\hat{\underline{c}}$ are within a tolerance of 10^{-10}. It is then necessary to threshold $\hat{\underline{c}}$. As in the detection step, we choose to threshold $\hat{\underline{c}}$ with a threshold parameter μ' set by taking a percentage of its ℓ^2 norm

$$\mu' = \rho'\|\underline{c}_T\|_2,$$

where ρ' is the percentage. From experiments, we choose $\rho' = 0.1$. We note the thresholded vector $\hat{\underline{c}}_{t_{\mu'}}$. The nonzero entries of the solution $\hat{\underline{c}}_{t_{\mu'}}$ determine the different events in $\underline{s}$. In particular, each nonzero entry in $\hat{\underline{c}}_{t_{\mu'}}$ corresponds to a particular atom φ_α in $\mathcal{D}$, whose parameter α gives the fractional exponent of the corresponding event. In the case of dictionaries parameterized by fractional exponents and frequencies, the nonzero entries in $\hat{\underline{c}}_{t_{\mu'}}$ correspond to atoms $\varphi_{(\alpha,f)}$, whose parameters α and f give the fractional-orders and frequencies of the estimated events.

4.4 Numerical scheme

We present the numerical scheme for our algorithm. We use the abbreviation BCR (Block Coordinate Relaxation) for the iterative thresholding algorithm, as our thresholding algorithm is based on the Block Coordinate Relaxation Method [24, 26]. T_μ denotes the hard thresholding

operator by μ. The function **Estimate** approximates the fractional-orders of the different windowed signals $\underline{s}_i$. The function **Global** gathers the estimated fractional-orders of the windowed signals into a global vector $\underline{a}$. The vector $\underline{a}$ contains the fractional-orders of the different events in $\underline{s}$.

The algorithm is given below:

DETECTION-ESTIMATION ALGORITHM

Build the detection dictionary $\mathcal{D}_d$
Build the estimation dictionary $\mathcal{D}_e$

- Perform for each trace $\underline{s}$

  DETECTION

  1. Decompose $\underline{s}$ with respect to $\mathcal{D}_d$ using Iterative Thresholding: $\underline{c} = \mathbf{BCR}(\underline{s})$
  2. Threshold $\underline{c}$ with threshold level μ: $\underline{c}_{t_\mu} = \mathrm{T}_\mu(\underline{c})$
  3. Select the p sets of largest coefficients: $(\underline{c}^i_{t_\mu})_{i=1}^p$
  4. Window out the p detected parts in $\underline{s}$: $(\underline{s}_i)_{i=1}^p$ corresponding to $(\underline{c}^i_{t_\mu})_{i=1}^p$

  ESTIMATION

  - For each part $\underline{s}_i$
    1. Decompose $\underline{s}_i$ with respect to $\mathcal{D}_e$: $\underline{c}_i = \mathbf{BCR}(\underline{s}_i)$
    2. Select the waveforms and estimate their fractional-order: $\underline{a}_i = \mathbf{Estimate}(\underline{c}_i)$
  - end

- Global vector for fractional-orders $\underline{a} = \mathbf{Global}(\underline{a}_i)$
- end

Chapter 5
Numerical Experiments

We now present the results of our detection-estimation method on both synthetic and real data. We consider dictionaries made with fractional derivatives of a centered Gaussian. The atoms of the dictionaries are thus parameterized by their fractional-order α and their standard deviation σ (equivalent to the scale and the frequency). The dictionaries can be written as

$$\mathcal{D} = (\varphi_{\alpha,\sigma})_{(\alpha,\sigma)\in(\mathbb{R}^+)^2},$$

where α and σ are the indexing fractional-orders and standard deviations. The fractional-order α of the waveform $\varphi_{\alpha,\sigma}$ is given by the fractional amount of differentiation α performed on the Gaussian:

$$\forall \alpha \geq 0 \ , \ \forall x \in \mathbb{R} \ , \ \varphi_{\alpha,\sigma}(x) = D^{\alpha} g_{\sigma}(x), \tag{5.1}$$

where g_σ is the centered Gaussian function with standard deviation σ, defined by $g_\sigma(x) = \mathrm{e}^{-\frac{x^2}{2\sigma^2}}$, for $x \in \mathbb{R}$. Our choice of dictionary made with fractional derivatives of the Gaussian is motivated by several factors. First, dictionaries made with fractional derivatives of the Gaussian provide flexibility over the choice of both the fractional-orders and the frequencies of the atoms. They hence provide adaptable dictionaries and in particular, allow for the construction of bandwidth limited dictionaries containing atoms with very localized frequencies. Bandwidth limited dictionaries are particularly well suited for seismic data, as seismic data is bandwidth limited. Second, we noticed that fractional derivatives of the Gaussian showed good correlation with seismic data, both in the time and frequency domain. In particular, their Fourier spectrum fits quite well the Fourier spectrum of seismic data. We show an example of the Fourier fitting

in Figure 5.1.

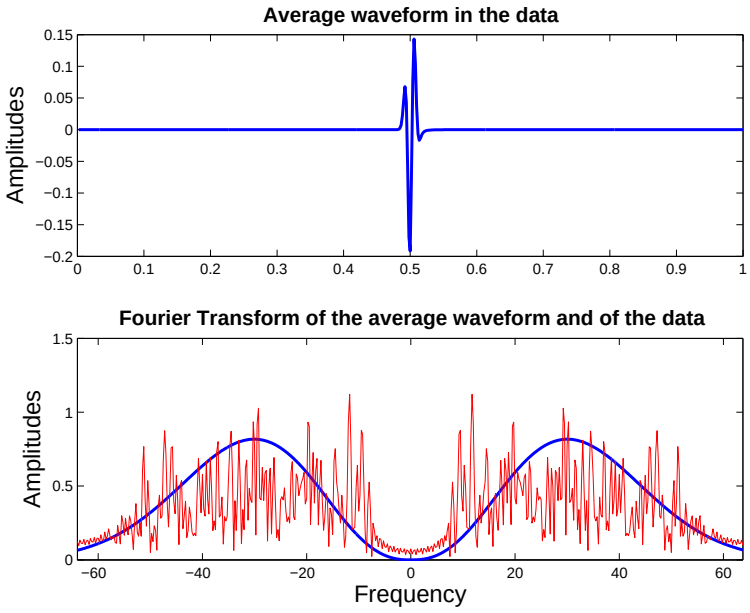

Figure 5.1: The top plot shows the average waveform in the data that was found by fitting the Fourier spectrum of the data with the Fourier spectrum of a fractional derivative of the Gaussian. The bottom plot shows the Fourier transform of the data (in red) and the average waveform (in blue).

The results displayed in Figure 5.1 have been made on real seismic data gratefully provided by ChevronTexaco. The top plot shows the average waveform of the data, whereas the bottom plot displays the superposition of the Fourier Transform of the average waveform and the Fourier Transform of the first trace of the data.

5.1 Application to synthetic data

5.1.1 Simple 1D synthetic data

We start by presenting the results of our detection-estimation algorithm on simple synthetic signals. We consider synthetic signals s as superpositions of fractional derivatives of the Gaussian:

$$s = \sum_{j=1}^{N_\sigma} \sum_{i=1}^{N_\alpha} c_k \varphi_{\alpha_i,\sigma_j}, \tag{5.2}$$

where $k = i + (j-1)N_\alpha$, and the waveforms $\varphi_{\alpha_i,\sigma_i}$ are taken from an overcomplete synthesis

dictionary $\mathcal{D}_s = (\varphi_{\alpha,\sigma})_{(\alpha,\sigma)\in(\mathbb{R}^+)^2}$. The synthesis dictionary refers to the dictionary we use to synthesize a signal s.

We apply our detection-estimation algorithm to two different synthetic signals: one with 20 events ($N = 20$), and one with 30 events ($N = 30$). We choose a synthesis dictionary $\mathcal{D}_s$ with 30 different fractional-orders α ranging from 1 to 4, and 25 standard deviations between 0.007 and 0.013. We choose $\lambda_{min} = 10^{-3}$ for the iterative thresholding algorithm of the detection part, and $\lambda_{min} = 10^{-9}$ for the iterative thresholding algorithm of the estimation part. The detection decomposition is performed with a dictionary of size 3, and the estimation decomposition with a dictionary of size 48: the estimation dictionary contains 16 different fractional-orders between 1 and 4, and 3 different standard deviations. The frequencies of the waveforms contained in the estimation dictionary can be chosen to be close to the frequency of the windowed event, which makes the estimation dictionary *scale-* or *frequency-*adaptable. This property of our algorithm enables us to reduce the number of standard deviations, and hence the size of the estimation dictionary. This reduction increases the speed of the estimation decompositions performed on each windowed event. We use this adaptability property in the analysis of the two synthetic signals.

The results obtained with our detection-estimation algorithm are shown in Figures 5.2 and 5.3.

Figure 5.2a shows the signal with 20 events, Figure 5.2b presents the original fractional exponents at their locations in the signal, and Figure 5.2c shows the fractional exponents found by the algorithm. In this simple example, the locations and fractional-orders of the different events in $\underline{s}$ are correctly found .

Figure 5.3a shows the synthetic signal s containing 30 events, Figure 5.3b shows the original fractional exponents α at their location in the signal, and Figure 5.3c shows the fractional exponents found by the algorithm. All the events in $\underline{s}$ are found and have the correct locations, and most of the fractional-orders are correct. However, some of the fractional-orders found by the algorithm are far from the original ones. We explain these few errors in the estimated fractional-orders to be due to an erroneous windowing. An incorrect windowing will not capture

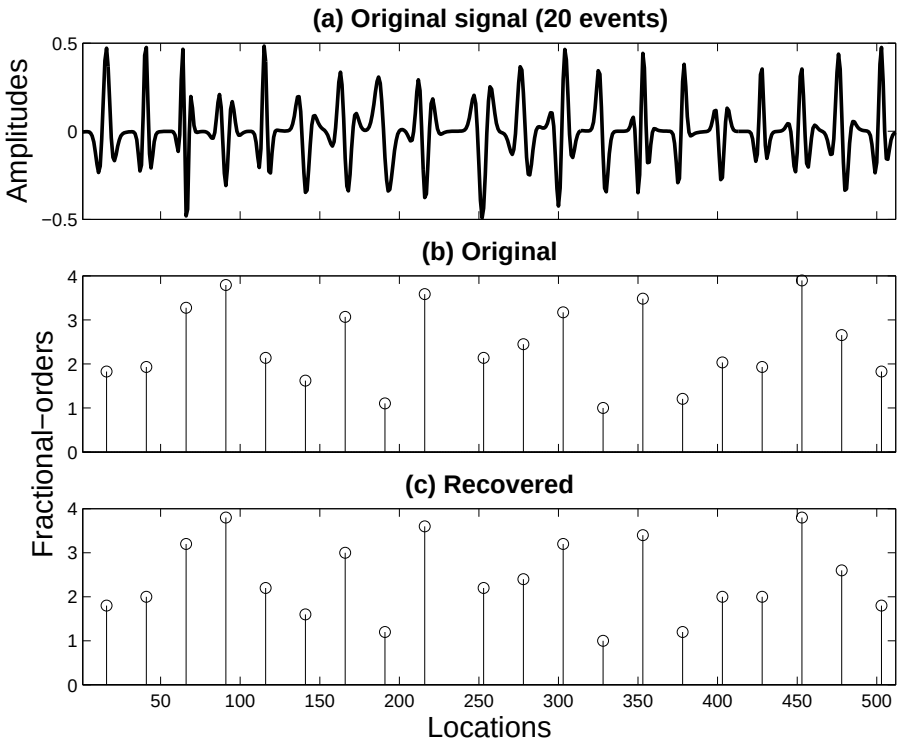

Figure 5.2: Analysis of a synthetic signal with 20 events.

the whole event and therefore not extract it entirely, providing the estimation algorithm with the wrong waveform. A poor windowing is more likely to happen when the signal $\underline{s}$ contains closely-spaced events relative to the scale of the seismic wavelet.

5.1.2 Comparison with Basis Pursuit and Matching Pursuit

In this section, we compare the performance of our detection-estimation algorithm with the performance of Basis Pursuit (BP) and Matching Pursuit (MP). We consider a signal s with 25 events ($N = 25$) and synthesize it with a synthesis dictionary $\mathcal{D}_s$ made with 30 fractional-orders α between 1 and 4, and 17 standard deviations between 0.09 and 0.011. We set the parameters of our algorithm to be $\lambda_{min} = 10^{-3}$ for the iterative thresholding algorithm of the detection part, and $\lambda_{min} = 10^{-9}$ for the iterative thresholding algorithm of the estimation part. We choose a detection dictionary of size 3 and an estimation dictionary of size 54. In this experiment we do not use the adaptability property of the algorithm in order to have a fixed estimation dictionary for all windowed event. In that way, we can use the estimation dictionary

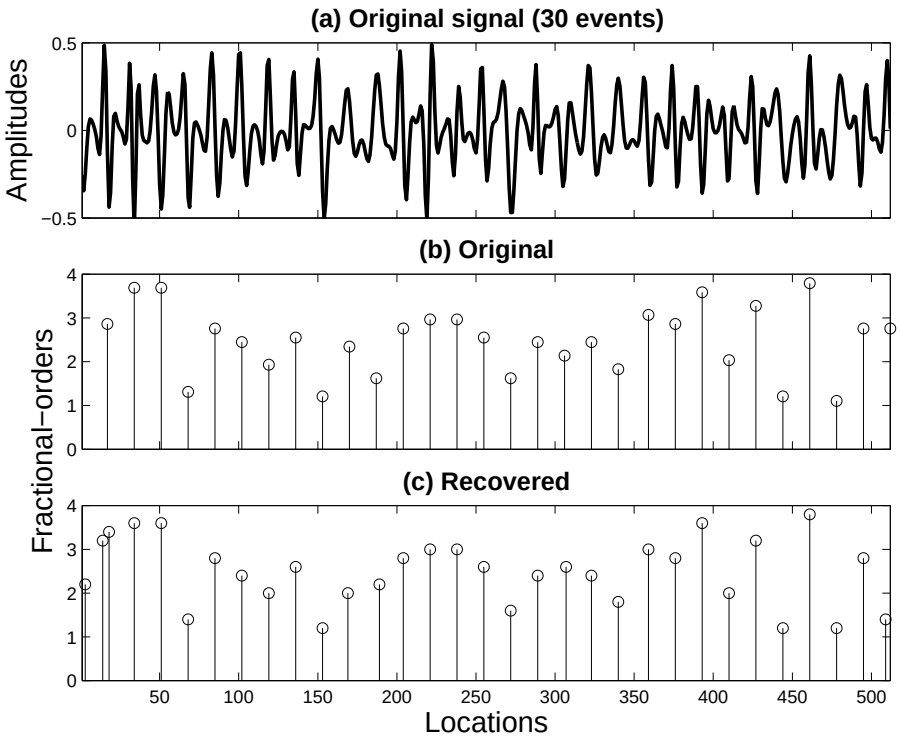

Figure 5.3: Analysis of a synthetic signal with 30 events.

as the analysis dictionary for BP and MP.

Both BP and MP decompose the signal s in an analysis dictionary $\mathcal{D}_a$, and try to approximate the sparsest representation $\underline{c}$, such that, $\underline{s} = \mathbf{\Phi}_a \underline{c}$, where $\mathbf{\Phi}_a$ is the matrix of the dictionary $\mathcal{D}_a$. We compare the results of our algorithm with the results of BP and MP, and use the estimation dictionary of our method as the analysis dictionary $\mathcal{D}_a$ for BP and MP. Each algorithm gives a representation of the signal s in the dictionary $\mathbf{\Phi}_a$. For simple signals and small dictionaries, MP, BP and SEEDE gives the same representation. However, when the signal complexity is high and the dictionary large, each algorithm provides different representations of the same signal s. The results of our comparison are shown in Figure 5.4.

The first top plot shows the original fractional exponents at their location in the signal s, the second top plot shows the fractional exponents found by our algorithm, that we call SEEDE (Scale Exponent Estimation by Detection-Estimation) for simplicity, the third plot shows the fractional exponents found by BP and the bottom plot shows the fractional exponents found by

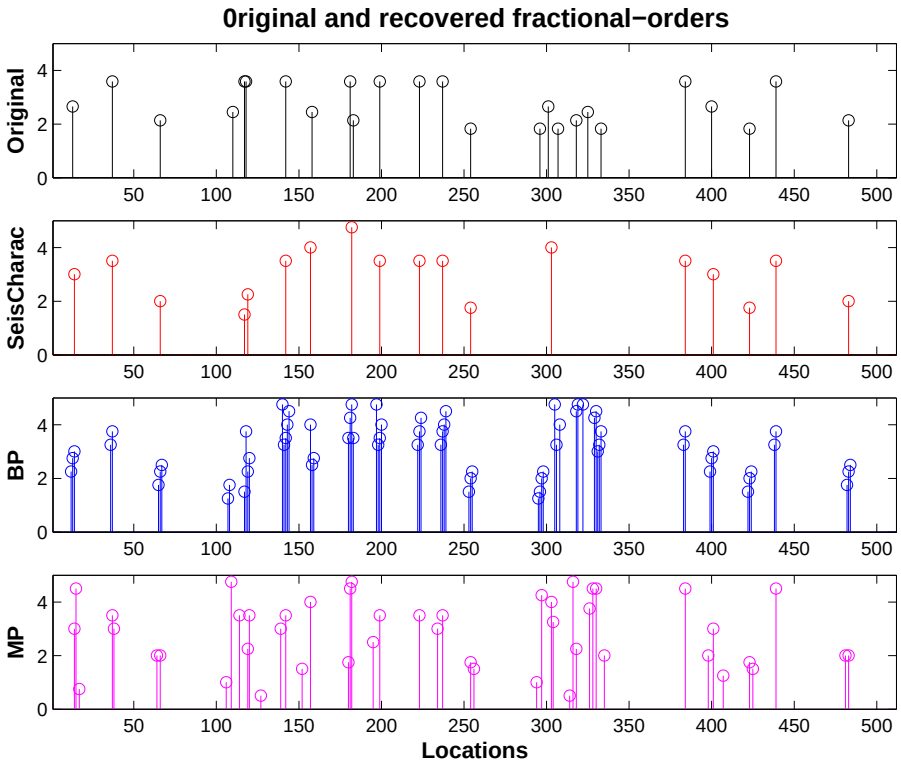

Figure 5.4: Comparison between the detection-estimation algorithm, BP and MP

MP. We can see that for this simple signal s, BP and MP globally find the correct events at the correct locations but find other events as well. BP still find more events than MP because of its superresolution property [2]. However, neither BP nor MP find the events located at abscissa 300 because too many events are closely-spaced in the vector.

For each event in the original vector, BP and MP also find multiple events with different fractional-orders. BP is nevertheless better than MP, because for most of the events in the original vector,and among the multiple fractional-orders found by BP, there is often the correct fractional-order.

Unlike BP and MP, our method doesn't resolve closely-spaced events, and its resolution property is lower than the resolution property of BP and MP. However, for each recovered event, our method finds only one fractional-order. We can see that most of the fractional-orders recovered by our method are correct. The incorrect fractional-orders are those belonging to events influenced by other neighboring events. This is due to the windowing which extracts a part of the signal, and prevents the estimation analysis to "see" the complete signal. Information is therefore "cut" by the windowing. As long as the events are not too closely-spaced, and no matter how many events the signal contain, our method will find and correctly estimate all the fractional-orders. However, as soon as the events start to interfere with each other, our method starts to have some difficulties in finding the correct fractional-orders and then the events themselves. One great advantage of our method compared too MP and BP, is that it gives only one estimated fractional-order for each event found. We feel that further work is required to get an optimal windowing and improve the resolution of our algorithm.

The numerical complexities of BP, MP and our method are quite different. Table 5.1 displays the CPU times in seconds spent in running different decomposition techniques on signals with various lengths. All computation was done on a Macintosh G5 workstation. For simplicity in the table, we denote our method SEEDE. MP is the fastest decomposition. MP has a quasi-linear complexity, depending on the number of chosen atoms [2, 21]. BP on the other hand, is very slow. The complexity of BP depends on the complexity of the conjugate gradient solver and

Problem Size: 512	CPU Running Time in Seconds		
Dictionary Size	BP	MP	SEEDE
18	22630.68	7.1450	10.4283
30	66605.47	11.0123	33.4668
42	5902.68	9.4068	56.5617
54	44589.83	12.4648	71.5451

Table 5.1: CPU Running Times of BP, MP and our method SEEDE

the complexity of the primal-dual logarithmic barrier interior-point algorithm. The complexity of BP is also affected by the size of the dictionary and the complexity of the signal. The numerical complexity of BP can nonetheless be modified by using different parameter settings. Our algorithm finds a compromise between BP and MP.

One should note that all our experiments were done in Matlab. The speed of these algorithms can therefore be largely increased by using C_{++} or Fortran.

5.1.3 Realistic synthetic data

We now present our result on more realistic synthetic data. Seismic data is usually stored in seismic cubes or a seismic matrices, where each column is a seismic trace. We use synthetic data of size 512×100. Figures 5.5, 5.7 and 5.8 present the results of our algorithm on a synthetic slice (or matrix) with 16 events.

Figure 5.5 displays the synthetic slice, Figure 5.6 shows the detected reflectors, Figure 5.7 presents the original fractional-orders at their location in the different traces. Figure 5.8 shows the fractional-orders found by our algorithm. We used a synthesis dictionary made with 20 different fractional-orders ranging from 1 to 4, and 3 standard deviations (scales) between 0.005 and 0.04. In the algorithm, we used a detection dictionary of size 2 (2 standard deviations) made with the average waveform estimated from the synthetic data. We chose the adaptive estimation dictionary.

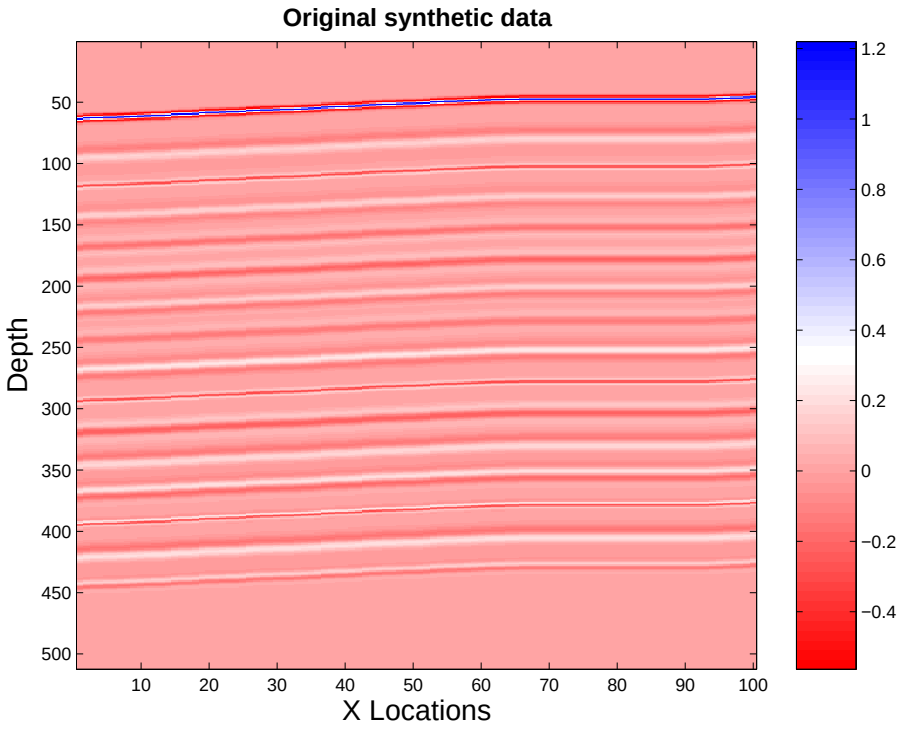

Figure 5.5: Synthetic slice with 16 reflectors.

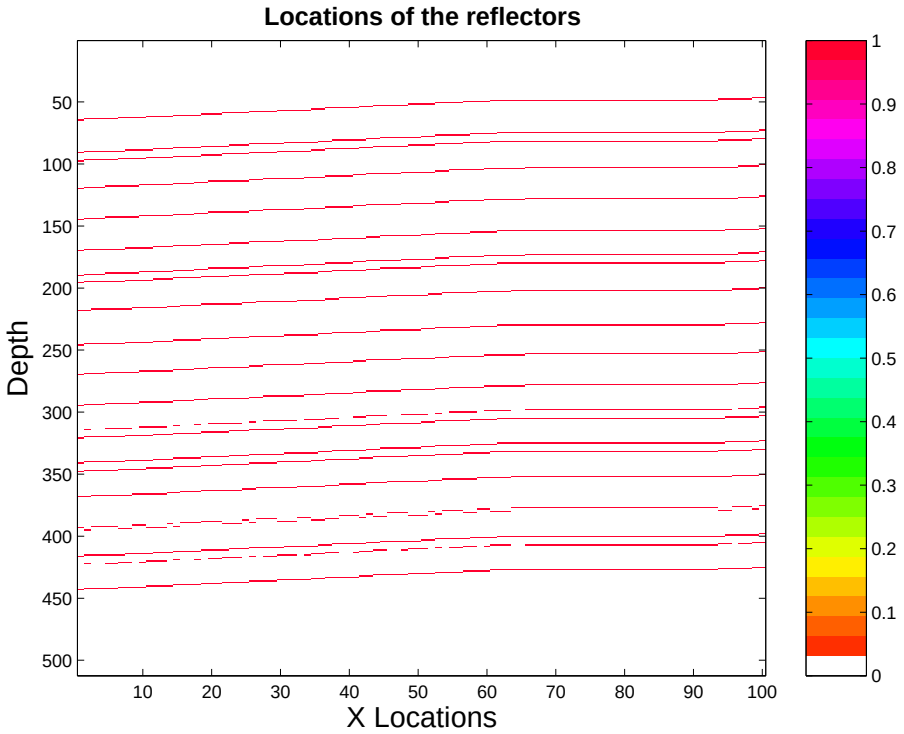

Figure 5.6: Detected reflectors

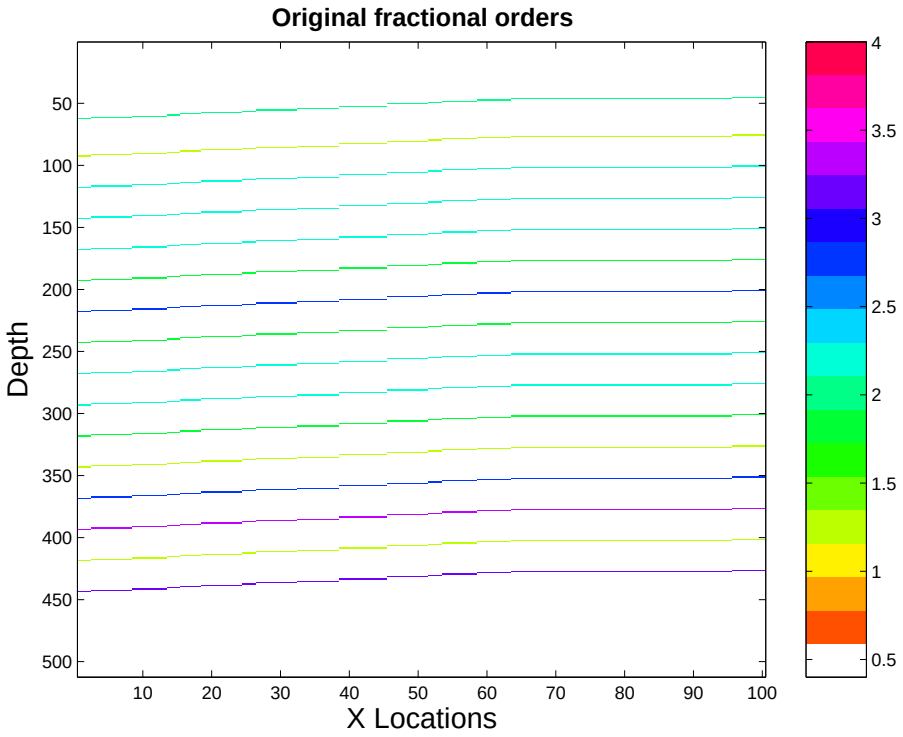

Figure 5.7: Original fractional-orders

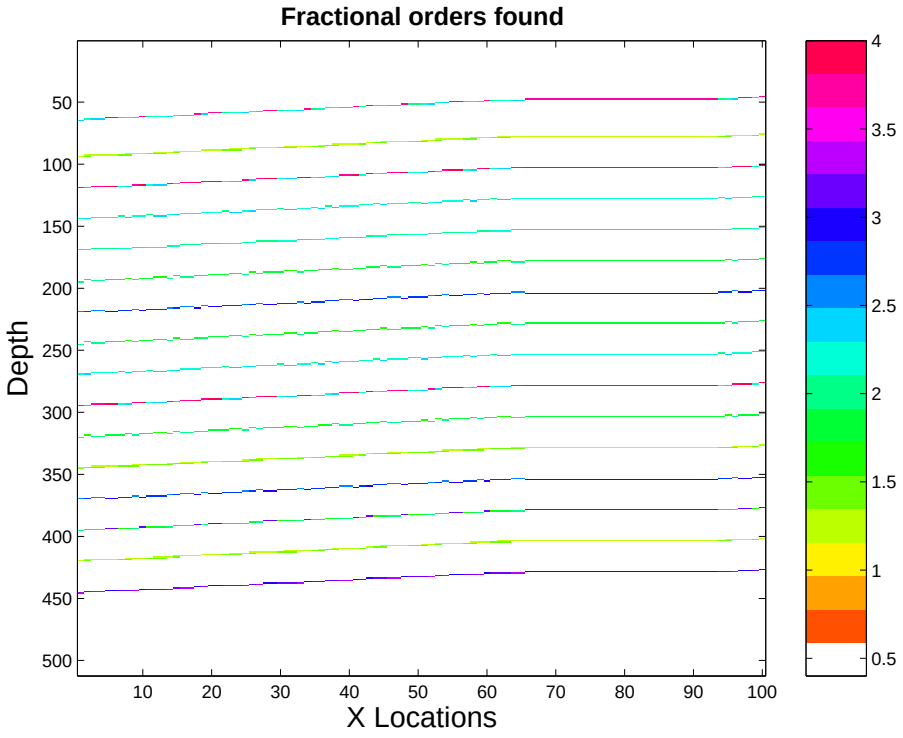

Figure 5.8: Recovered fractional-orders

In this example, the event are quite well separated so the algorithm can detect them. This example shows that if the events are well separated, the algorithm detects all of them and correctly estimates most of them. The lateral continuity of the recovered fractional-orders along the different reflectors shown in Figure 5.8 is overall quite well preserved. This property is very important for seismic data, as fractional-orders should remain laterally continuous along reflectors. However, although most of the recovered fractional-orders recovered are correct and lateral continuous along the reflectors, we can clearly see that three distinct reflectors have fluctuating fractional-orders. The fractional-order oscillated between the correct value and the maximum fractional-order of the dictionary. This phenomenon is due to an incorrect estimation of the standard deviation in the detection part. The frequency of the analyzed event in the estimation part, is indeed very important for a correct estimation. If the frequency of the atom is incorrect, the estimation will be unable to find the correct fractional-order as the largest inner product between the event and the atoms in the estimation dictionary will not correspond to the atom with the correct fractional-order. In this example, we clearly see that sometimes, the detection outputs an incorrect standard deviation. The reason for that happening is not well understood at present, and we feel that further work is required to fully understand the whole algorithm.

We now show the results of our algorithm on a different synthetic slice containing 24 events.

In Figures 5.9, 5.10, 5.11 and 5.12, we show the results of our algorithm on a synthetic slice containing 24 events. Figure 5.9 shows the synthetic data, Figure 5.10 shows the output of the detected reflectors, Figure 5.11 displays the original fractional-orders of the different reflectors, and Figure 5.12 presents the fractional-orders found by our algorithm. Like the previous example, we used a synthesis dictionary made with 20 fractional-orders between 1 and 4, and 3 standard deviations between 0.005 and 0.04. In the algorithm, we used a detection dictionary of size 3 containing 3 different standard deviations, and made with the average waveform. We chose the adaptive estimation dictionary.

In this example, the events start interfering with each other, although not extremely close. One

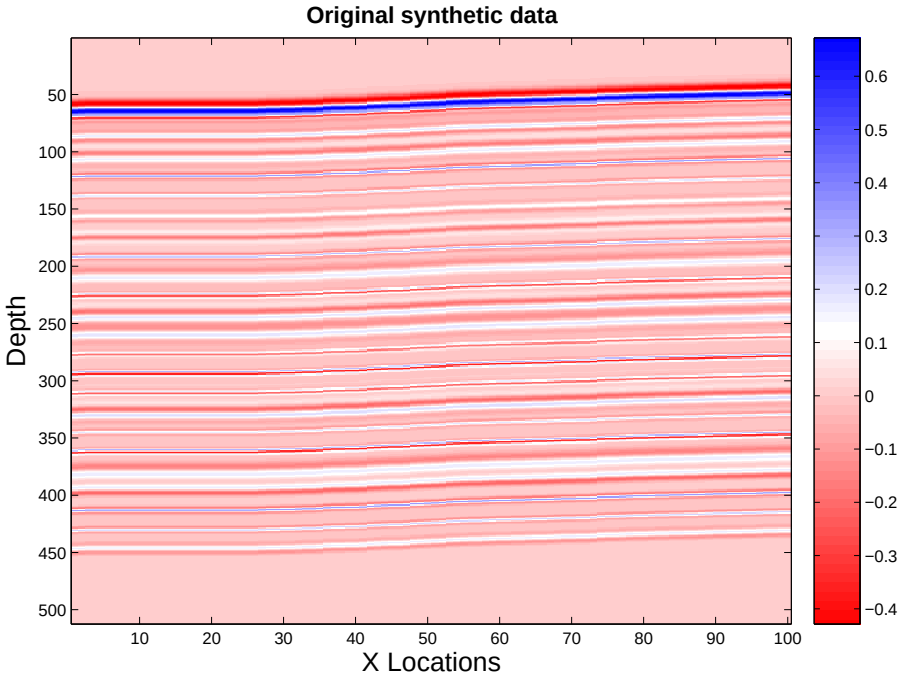

Figure 5.9: Synthetic slice with 24 reflectors

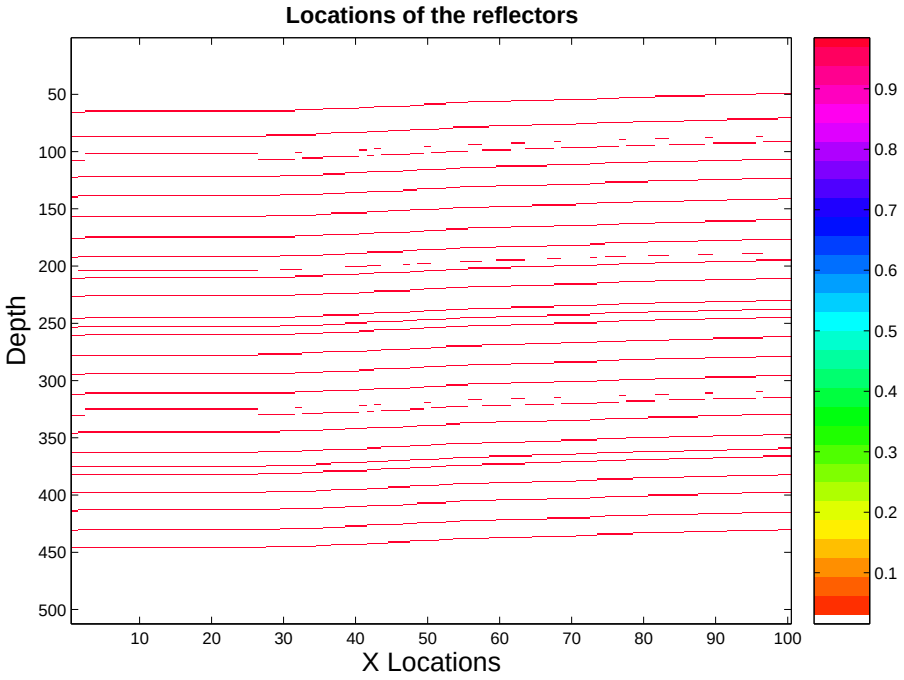

Figure 5.10: Detected reflectors

can clearly notice that the second event was not found by the algorithm, because it was too close to the first one. The algorithm lacks the superresolution property of BP and does not resolver two closely-spaced events. The two first reflectors represent a thin layer in the subsurface, but the algorithm considers it as only one reflector. In theory, this is not a problem, as the estimation algorithm should be able to detect closely-spaced events as well. However, due to numerical complexities, the current implementation of the algorithm does not yet resolve close-spaced events. Incorporating the superresolution property in our implementation of our algorithm is still ongoing research. Oscillating fractional-orders along certain reflectors can also clearly be seen in this example. One of the solution would be to increase the size of the detection dictionary and incorporate many scales and possibly different fractional-orders. However,the use of large detection dictionaries leads to denser coefficients that are hard to analyze.

However, despite these flaws, we can see that most of the fractional-orders found by our algorithm have the correct values, and remain overall laterally continuous along the different reflectors. Further work is still required to precisely understand the role of the different parameters involved in the algorithm. The algorithm can be largely improved by the adequate choice of dictionaries, minimization parameters and efficient minimization solver. The choice of the iterative thresholding algorithm is a first step in the implementation of our method, and will be replaced by more efficient solvers in the future.

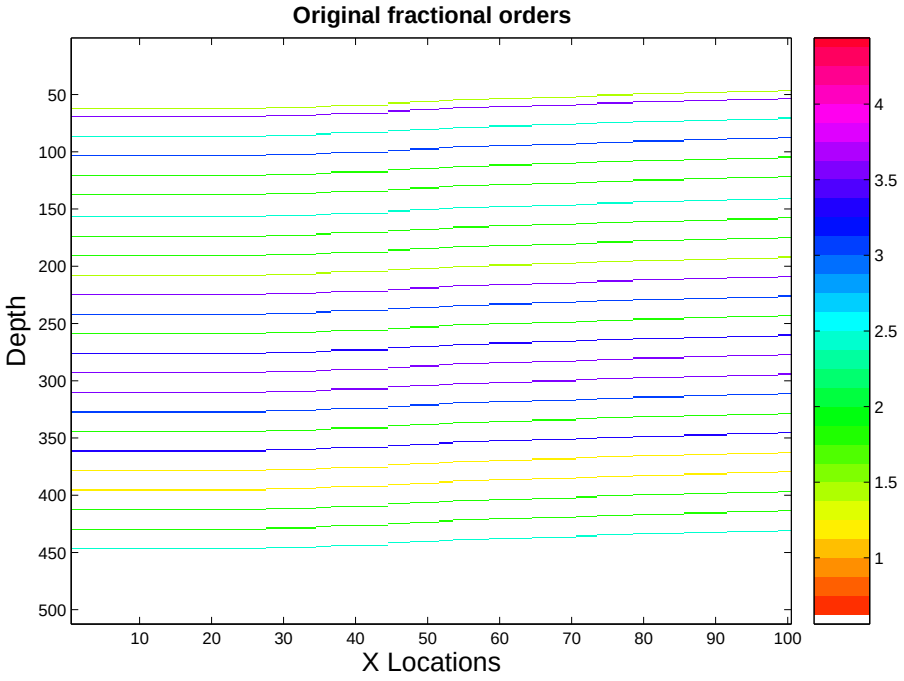

Figure 5.11: Original fractional-orders

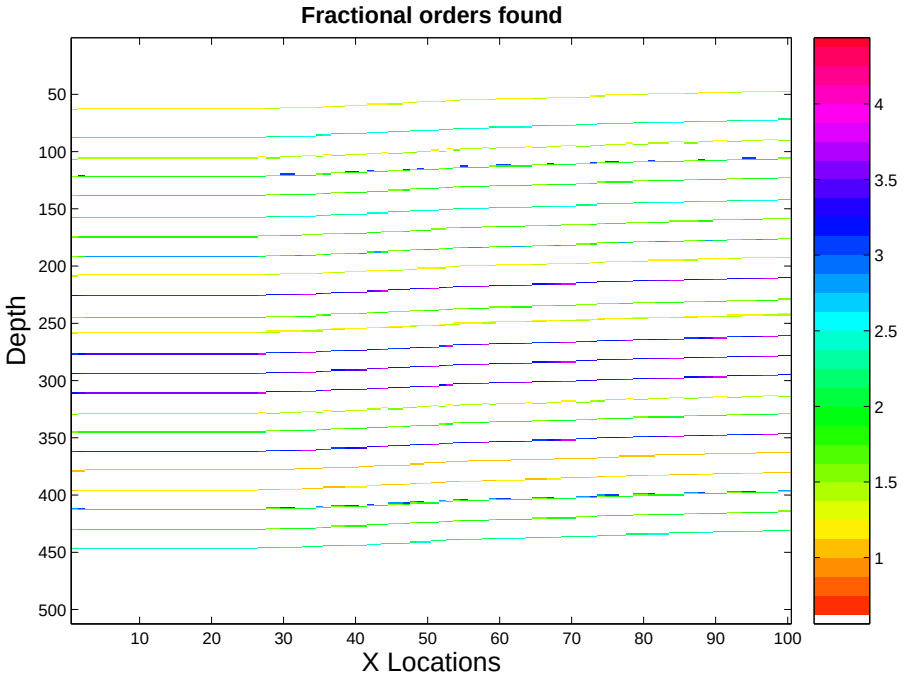

Figure 5.12: Recovered fractional-orders

5.2 Application to real data

We now present the results of our algorithm on real seismic data. We used a detection dictionary of size 1 made with the average waveform estimated through a Fourier spectrum fitting. The Fourier spectrum fitting was shown in Figure 5.1. We chose a detection dictionary of size 1 made with the average waveform in order to have a sparser and clearer detection coefficient $\underline{c}$. As stated in section 5.1.3, the larger the detection dictionary, the denser the output coefficient of the detection minimization, which makes it hard to detect the events and therefore defeats the purpose of the detection. In theory and for very clean data, the larger the dictionary, the better as larger dictionaries give more information than smaller dictionaries. However, in practice and for real data, the output of the detection algorithm gives a noisy coefficient that is difficult to analyze. We noticed from experiments on real data, that the efficiency of the detection algorithm is increased by using a smaller dictionary. The optimal performance of the detection algorithm was obtained with a detection dictionary of size 1. We used an estimation dictionary of size 54, with 18 fractional-orders, and 3 standard deviations. The fractional-orders and standard deviations of the estimation dictionary are chosen according to the fractional-order and standard deviation of the average waveform. In that sense, both the detection and estimation dictionaries are adaptable to the data. In the detection minimization, we used $\lambda_{min} = 10^{-4}$, and $\lambda_{min} = 10^{-10}$ for the estimation minimization.

We present the result of our algorithm in Figures 5.13, 5.14 and 5.15.

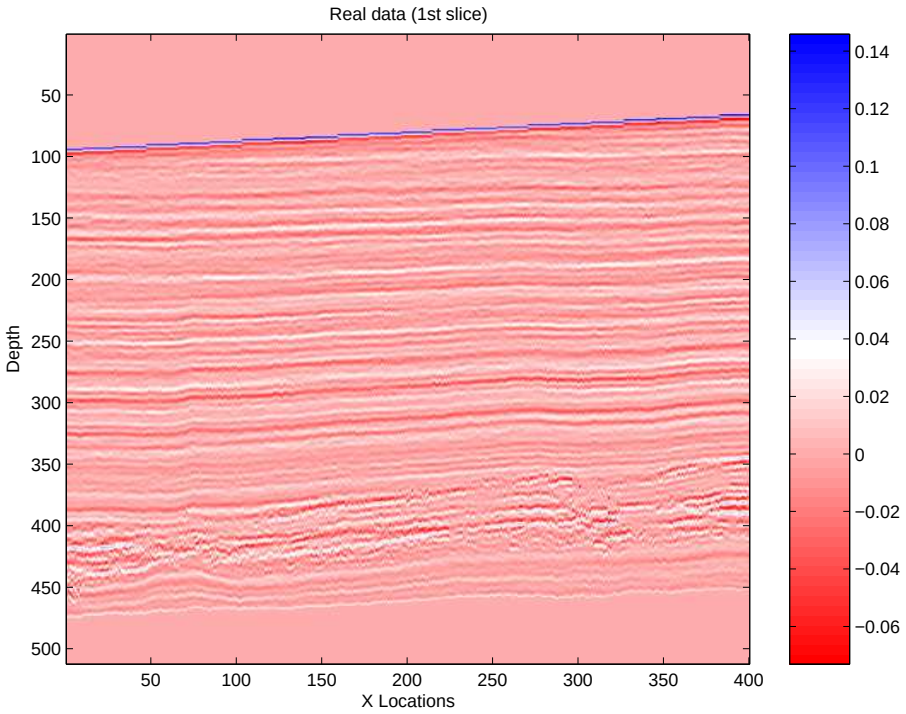

Figure 5.13: Real data

Figure 5.13 shows the first slice of a real seismic data cube, Figure 5.14 presents the detected reflectors, and Figure 5.15 displays the results of our algorithm on the first slice of the real seismic data. In Figure 5.15, we can clearly distinguish most of the different reflectors contained in the data, which shows a good performance of the detection algorithm. Some reflectors remain hard to discern, in particular at the bottom of the seismic slice, which is also the case in the real data shown in Figure 5.13. Overall, the results of our algorithm is coherent with the real data, and the reflectors detected follow the reflectors present in the real data. The lateral continuity of the fractional-orders along the different reflectors is quite well preserved for certain reflectors that are isolated and easy to detect. For general setting of reflectors, we can notice some fluctuations in the fractional-orders, which can be due to a wrong estimation in the frequency of the detected event, which is the phenomenon mentioned earlier, and/or to the choice of fractional-orders in the estimation dictionary. The changes in the fractional-orders can also be due to actual changes in the nature of the transitions in the subsurface.

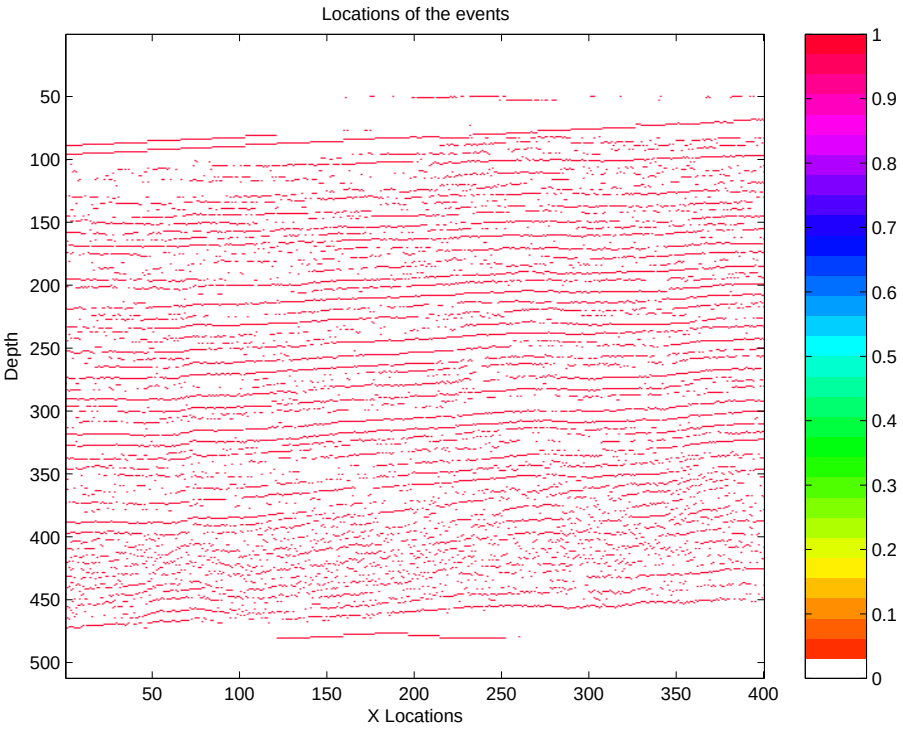

Figure 5.14: Detected reflectors

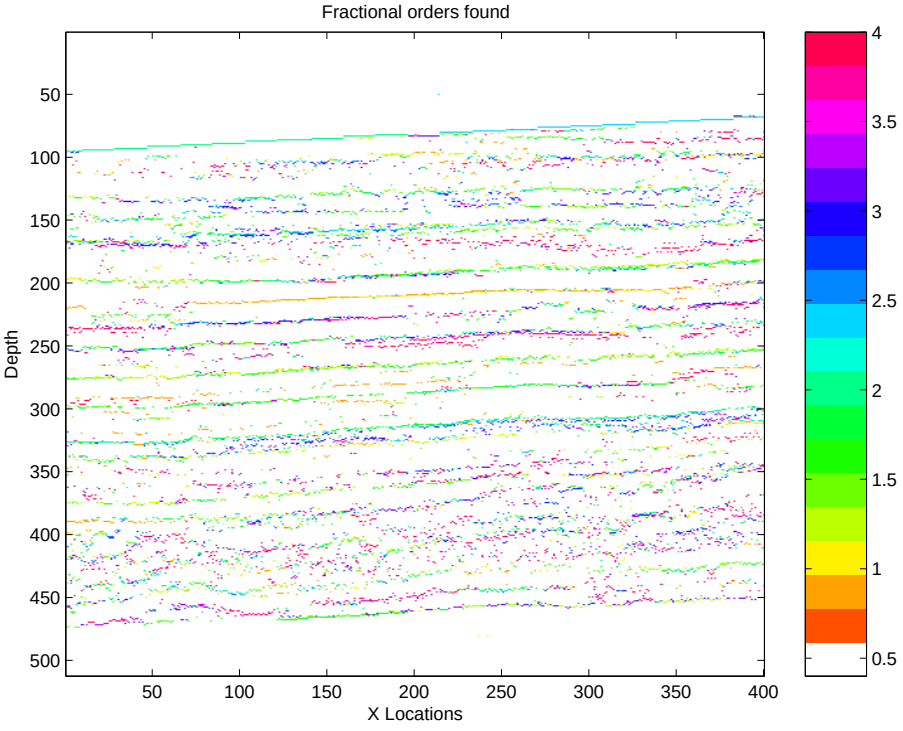

Figure 5.15: Recovered fractional-orders

Despite limitations in our current implementation of the method, we feel that our preliminary results are encouraging with respect to detecting the reflectors and estimating their fractional-orders.

Figures 5.16 presents the superposition of the real data with the fractional-orders recovered by our algorithm.

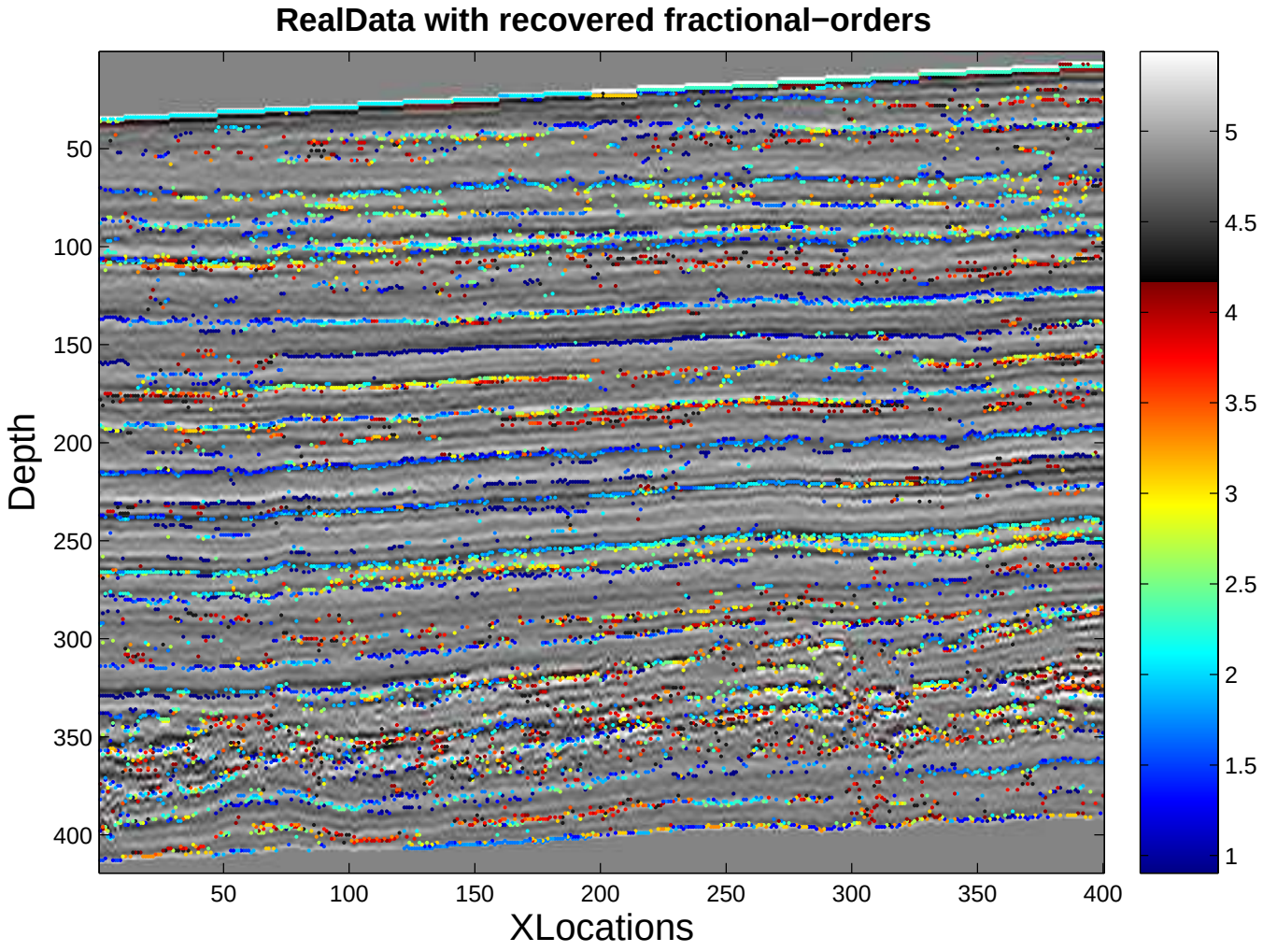

Figure 5.16: The real data is presented by the grey scale and the recovered fractional-orders are displayed in colors.

Chapter 6
Discussion

6.1 Summary and Conclusions

In this paper, we have presented a new method for characterizing and delineating transitions in the subsurface. Our method is based on optimal decompositions of the data [2], with respect to overcomplete dictionaries of waveforms parameterized by their fractional-orders. We propose a two-step approach that divides the initial ℓ^1 decomposition problem into a detection step followed by an estimation step. The detection part locates the predominant events in the data, and partitions the data into its prevailing components. The estimation part approximates the fractional-orders of each detected component. The detection and estimation algorithms both exploit the sparsity property of the representation of the data with respect to the dictionaries, by minimizing the ℓ^1 norm of the data representation. We use an iterative thresholding algorithm [4] with a Block Coordinate Relaxation method [24, 25] to solve the minimization.

We show that by first partitioning the data, and then analyzing each detected event separately, we improve the performance of BP with respect to finding the original coefficient. The probability of success in finding the original representation of the data with respect to a given dictionary $\mathcal{D}$, is increased by our new technique. In particular, the probability of detecting the events in the data with our new method is quite large. We show that our method achieves quite good performance in detecting events, provided the events are not too closely-spaced. The method also gives encouraging results with respect to the estimation of the fractional-order of the events, and provide an increased probability of correctly detecting and estimating the event than BP. x

Unlike BP, where the use of large dictionaries is prohibitive, our new method can give good and accurate results in fairly large dictionaries. Besides, the computational complexity of our method can be reduced by changing the speed parameters of both the detection and estimation decomposition algorithms. However, the complexity of our algorithm is still significant, and increases with the size of the dictionaries. In comparison to BP, our method achieves lower computational cost, but is computationally more expensive than MP. In that respect, our method seems to give a compromise between MP and BP. In order to improve the resolution of our method, we proposed to replace the iterative thresholding algorithm of the estimation step by BP. However, incorporating BP into the estimation part made it prohibitively expensive when dealing with large dictionaries. We therefore feel that further work is still required to achieve superresolution of thin layers with our new method.

We propose a fast and efficient detection algorithm by using a small dictionary. The goal of the detection algorithm is to both detect the major events in the data and estimate their frequency through an approximate ℓ^1 decomposition in a small dictionary. The estimated frequency of the events is indeed very important, as it constitutes the base of the frequency contents of the estimation dictionary. In the case where the estimation dictionary contains the wrong frequencies, the largest inner product between the atoms in the dictionary and the analyzed event will pertain to an atom with an incorrect fractional-order. It is therefore crucial that the detection algorithm gives a good approximation of the correct frequency of the events. There is a trade-off between an optimal performance of the detection, which requires a small detection dictionary (especially for real and noisy data), and a good approximation of the frequency of the detected events, which requires a larger detection dictionary with a wide frequency content.

Contrary to the detection algorithm, the estimation algorithm uses very large dictionaries with a wide range of fractional-orders, which provides accuracy in the estimated fractional-orders. Unlike the detection decomposition, the estimation decomposition, due to the size of the dictionaries involved, is a slow and computationally intensive algorithm. However, many ways are possible to increase the speed of the estimation part, including changing the speed parameters in the iterative thresholding algorithm, and reducing the size of the estimation dictionary by

adapting its frequency to the frequency of the analyzed event. The estimation dictionary can therefore be adapted to the event by narrowing its frequency content and selecting only a few frequencies around the frequency estimated in the detection. In that case, if the detection provides an incorrect estimated frequency, the estimation will be unable to find the correct fractional-order. One should note that incorporating a very large frequency content in the estimation dictionary, in addition to its wide range of fractional-orders, leads to an extremely slow and computationally intensive algorithm.

The experiments performed on both synthetic and real data nonetheless show promising outcomes. The performance of our algorithm on synthetic data gives very encouraging results as the lateral continuity along the different reflectors is quite well preserved. The results on real data, although lacking some lateral continuity along reflectors, seem also quite promising.

6.2 Limitations of our method

The method suffers from a limitation in the resolution of two closely-spaced events in the data. At the current stage, the estimation algorithm doesn't resolve two-closely spaced events. The method therefore loses the superresolution property of BP, but gains in lower computational costs.

Another limitation is the large dependence between the detection algorithm and the estimation algorithm. A correct approximation of the frequencies of the detected events is a very important element for the estimation algorithm. The approximation of the frequency of the detected events indeed constitutes the baseline for the frequency content of the estimation dictionary. Besides, there is a large trade-off between the size of the detection dictionary, which should contain a wide range of frequency to ensure a correct approximation of the frequency of the detected events, and the efficiency of the detection algorithm. The wider the frequency content of the detection dictionary, the more accurate the frequency of the detected events, but the less efficient the detection of the events, in particular for real and noisy data. The analysis of the coefficients obtained in the detection algorithm therefore needs to be improved in order to

correctly detect the events in large dictionaries with wide frequency content.

Although our method achieves a good performance in fairly large dictionaries, its efficiency is still limited by the size of the dictionaries, due to computational costs. The estimation dictionary cannot incorporate a wide range of frequencies, as its size would lead to a prohibitively slow algorithm.

The windowing of the detected events is also a very sensitive part that plays an important role in the estimation algorithm, as it determines the efficiency of the estimation. A wrong windowing can lead to the extraction of an incorrect event, and therefore to an incorrect fractional-order. If the window is too small, the event will be "cut", and not extracted entirely. On the other hand, if the window is too large, the extracted event will contain remainings of other events. In those cases, depending on how much the window "cut" the event, or on how much the window added to the event, the estimated fractional-order will be correct or incorrect. Further work is still required to build an optimal windowing.

Even with these limitations, our method still performs quite well in the detection of the events, and relatively well in the estimation of their fractional-orders. In particular, the results obtained on real data, showed that most of the reflectors were detected by our algorithm, and that the lateral continuity of the fractional-orders along the reflectors was relatively well preserved, taking into account that actual changes in the fractional-orders of the reflectors also exist in the data. The goal of our algorithm is to detect these changes, and we feel that our algorithm performed quite well with respect to the smooth changes in the fractional-orders of the reflectors in the data.

6.3 Future work

In this paper, we developed a new method that demonstrated encouraging results, in particular in the detection part. It also showed that, no matter how many events were in data, as long as they were not too closely-spaced, our algorithm would detect them, and estimate them quite accurately.

In the future, it would be beneficial to improve the algorithm by making it faster and more efficient. The possible improvements are listed and explained below:

1. The current method uses an iterative thresholding algorithm with a Block Coordinate Relaxation method to solve an ℓ^1 minimization which enables us to use relatively large dictionaries. However, iterative minimizations lead to slow algorithms and therefore limit their performance. The numerical complexity of our algorithm, although inferior to the numerical complexity of BP, is still significant. It would therefore be good to increase the speed of the iterative thresholding solver, and to incorporate, in the future, more efficient ℓ^1 solvers involving global optimization tools.

2. Working with large datasets and large dictionaries involves numerous and costly computations, and therefore requires powerful programming languages. Our algorithm, currently written in Matlab, would gain in speed if it were programmed in another programming language like C_{++} or Fortran. In the future, it will be necessary to convert our Matlab routines into one of these programming languages.

3. Resolving thin layers in the subsurface is also very important, and constitutes a major issue when analyzing seismic data. In order to achieve superresolution, it would be interesting to incorporate BP in our algorithm, but in a way that would not lead to a prohibitively slow algorithm. BP is a very powerful solver with respect to its multiresolution property, and we feel that it can help our algorithm resolve closely-spaced events.

4. The efficiency of our method highly depends on the choice of the dictionaries, in particular on the choice of their fractional-orders and frequencies. Many questions therefore arise:

   (a) What is the range of fractional-orders and frequencies?

   (b) How finely sampled should the fractional-orders and frequencies?

   (c) Can we reduce the range of fractional-orders to $[0, 1]$ and build any other fractional-order $\alpha > 1$ through a linear combination of fractional-orders in $[0, 1]$?

5. Global multiscale analysis with wavelets performed on seismic data (see section 2.2.2) yield an estimate for the singularity spectrum $f(\alpha)$ of the data. The singularity spectrum $f(\alpha)$ gives the probability of occurrence of a particular singularity ın the data. The singularity spectrum therefore provides information on the different fractional-orders contained in the data. It would be interesting to use this information as a weighting function for the coefficients obtained in the ℓ^1 minimizations in the algorithm. The ℓ^1 norm of the decomposition of the data with respect to a dictionary associated to a rare singularity should be highly sparse, and therefore should be largely penalized in the minimizations. On the other hand, the ℓ^1 norm of the decomposition of the data with respect to a dictionary associated to a frequent singularity, should be less sparse, and therefore should not be penalized in the minimizations. The weighting function w should behave like

$$w(\alpha) \approx \frac{1}{f(\alpha)},$$

where $f(\alpha)$ is the singularity spectrum of the data.

For a signal $\underline{s}$ of length n, and a dictionary $\mathcal{D}$ of cardinality p, the minimization should be:

$$\min_{\underline{c}} \|\underline{s} - \mathbf{\Phi}\underline{c}\|_2 + \lambda\|\underline{wc}\|_1 \tag{6.1}$$

where $\underline{w}$ is the weighting vector constructed according to the fractional-orders contained in the dictionary $\mathbf{\Phi}$, $\underline{c} \in \mathbb{R}^m$ with $m = np$, and $\underline{wc}$ denotes the componentwise multiplication: $\forall 1 \leq k \leq m$, $(\underline{wc})(k) = \underline{w}(k)\underline{c}(k)$. If we write the dictionary $\mathcal{D}$ as:

$$\mathbf{\Phi} = [\mathbf{\Phi}_{\boldsymbol{\alpha_1}}, ..., \mathbf{\Phi}_{\boldsymbol{\alpha_p}}], \tag{6.2}$$

$\underline{w}$ can be obtained by the following computation:

$$\underline{w} = [\frac{1}{f(\alpha_1)}1_n, ..., \frac{1}{f(\alpha_p)}1_n], \tag{6.3}$$

where $1_n = [1, ..., 1]$ is a vector of ones of length n, and $\underline{w} \in \mathbb{R}^m$ with $m = np$.

6. Because the estimation of the frequencies of the detected events is very important, it would be interesting to include in the algorithm another estimation decomposition that would

estimate the frequency of each detected event. The global algorithm would then consists of a detection, a first estimation for an approximation of the frequency of each detected event, and a second estimation directed at finding the fractional-orders of each detected event. The first estimation dictionary would contain a wide range of frequencies and a few fractional-orders, whereas the second estimation dictionary would contain a wide range of fractional-orders and a few frequency centered around the estimated frequency. The proposed algorithm aims at improving the estimation of the fractional-orders, but gains computational cost. Performing two estimations with two large dictionaries can lead to a prohibitively slow algorithm with a high numerical complexity. Further work is therefore required to lower its computational cost. Figure 6.1 presents the modified algorithm.

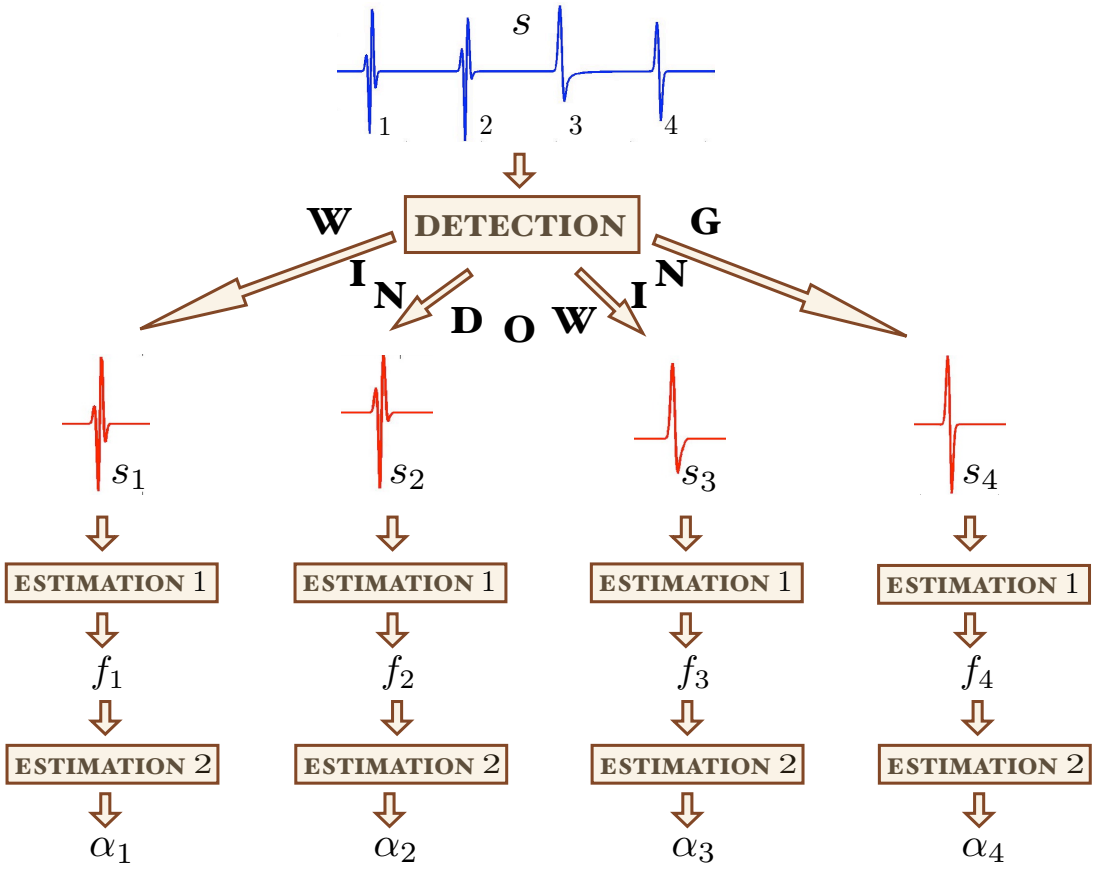

Figure 6.1: Improved detection-estimation method.

Part II

Capita Selecta

Chapter 7
Atomic decompositions with Redundant Dictionaries

7.1 Problem formulation

As stated in chapter 2 by equation (2.15), we model seismic signals s as

$$s(z) = \sum_{i \in \Lambda_C} C_i D^{-\alpha_i} \psi(z - z_i) \pm \sum_{i \in \Lambda_A} C_{\alpha_i} D^{-\alpha_i} \psi(z - z_i), \tag{7.1}$$

where z_i is the location of the ith transition in the subsurface, $C_i = \frac{\Gamma(-\alpha_i)K_i}{\Gamma(\alpha_i)} = \frac{\alpha_i a_i \Gamma(-\alpha_i)}{\Gamma(\alpha_i+1)}$, ψ is the seismic source function, and $D^{-\alpha_i}$ is the Liouville $-\alpha_i$th fractional derivative operator defined in definition 2.1.2. The goal is to estimate the locations z_i of the reflectors in the subsurface, their amplitudes C_i, and their fractional exponents α_i. For that, we consider seismic signals s as a superposition of waveforms ϕ_α parameterized by their fractional-order α:

$$s = \sum_{\alpha \in A} a_\alpha \phi_\alpha, \tag{7.2}$$

and envision their decomposition:

$$s = \sum_{\alpha \in S_0} c_\alpha \varphi_\alpha, \tag{7.3}$$

with respect to an overcomplete dictionary $\mathcal{D} = (\varphi_\alpha)_{\alpha \in S}$, where the atoms φ_α are waveforms parameterized by their fractional-order α, and S is the indexing set of the parameter α. In a matrix form, equation (7.3) can be written as:

$$\underline{\mathrm{s}} = \mathbf{\Phi} \underline{\mathrm{c}}, \tag{7.4}$$

where $\mathbf{\Phi}$ is the matrix of the dictionary $\mathcal{D}$. The goal is to find the representation $\underline{c}$ of the signal $\underline{s}$ which satisfies equation (7.4). However, due to the overcompleteness of $\mathcal{D}$, there is no unique solution for $\underline{c}$, but the nonuniqueness of $\underline{c}$ provides a flexibility which enables us to choose the most suited representation $\underline{c}$ to our problem. In the case of seismic signals, assumed to be a superposition of a limited number of waveforms, it is natural to try to minimize the number of components of $\underline{c}$.

There have been several approaches to solve this problem, including the Method of Frames [3], Matching Pursuit [21], Basis Pursuit [2] and the Block Coordinate Relaxation Method ([24, 25]). We describe those methods in the following sections.

7.2 Method of Frames

7.2.1 The algorithm

The Method of Frames [3] is an algorithm that finds the representation $\underline{c}$ with minimum ℓ^2 norm, and solves

$$\min_{\underline{c}} ||\underline{c}||_2 \text{ , subject to } \underline{s} = \mathbf{\Phi}\underline{c}. \tag{7.5}$$

(7.5) is a quadratic system whose unique solution $\underline{c}^\dagger$ is given by:

$$\underline{c}^\dagger = \mathbf{\Phi}^\dagger \underline{s} = \mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{\Phi}^T)^{-1}\underline{s}. \tag{7.6}$$

The matrix $\mathbf{\Phi}^\dagger$ is called the generalized inverse of $\mathbf{\Phi}$, and is given by: $\mathbf{\Phi}^\dagger = \mathbf{\Phi}^T(\mathbf{\Phi}\mathbf{\Phi}^T)^{-1}$. In our case, the major disadvantage of this method is the non sparsity of the solution $\underline{c}^\dagger$ [2]. The solution $\underline{c}^\dagger$ is indeed not sparse because the ℓ^2 norm tends to prefer a large number of small nonzero coefficients, rather than a few number of larger ones. The non sparsity of $\underline{c}^\dagger$ means that the decomposition of $\underline{s}$ is spread out on a large number of atoms in $\mathcal{D}$, which translates into the fact that we need a large number of atoms to represent $\underline{s}$, and contradicts

our previous assumption. The solution given by the Method of Frames is therefore not suited to our application.

7.2.2 Concept of sparsity

The concept of sparsity is intrinsically linked to the dictionary we use, as the decomposition of a given signal f can be sparse in one dictionary and not in another one.

Consider a simple example of a cosine function, denoted g. The representation of g in the Discrete Cosine dictionary (Discrete Cosine Transform) is very sparse, as we only need a few cosine basis functions to construct g. However, the representation of g in the Haar wavelet dictionary is full, as we need a large number of step functions to build g. The Haar wavelet basis is a basis made of translations and dilations of a step function. The results are shown in Figure 7.1.

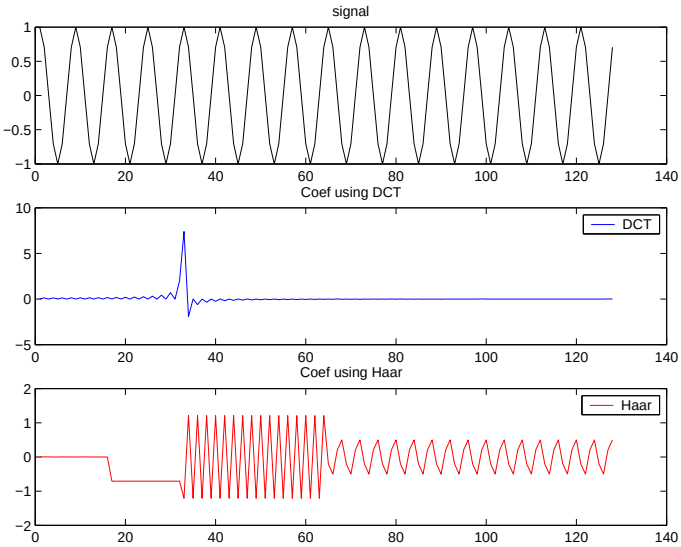

Figure 7.1: Decomposition of a cosine using the Discrete Cosine Transform (DCT) and the Haar wavelets.

The top figure shows the cosine function g, the middle one plots the decomposition of g with respect to the DCT dictionary, and the bottom one gives the decomposition of g with respect to the Haar basis.

7.3 Matching Pursuit

Contrary to the Method of Frames, Matching Pursuit (MP) [21] approximates the sparsest representation $\underline{c}_0$, where the sparsest representation refers to the representation with minimum ℓ^0 norm. MP tries to find the decomposition of the signal s in a dictionary $\mathcal{D}$ by iteratively finding the waveforms that best match the data s. MP is an iterative algorithm that selects, at each iteration, the waveform that minimizes the ℓ^2-difference between the subsequent reductions and their projection onto a waveform.

To describe the algorithm, we consider the discrete version of s, denoted $\underline{s}$. Consider a dictionary $\mathcal{D}$ made of atoms $\underline{g}_\lambda$. The Matching Pursuit algorithm begins by projecting $\underline{s}$ on a vector $\underline{g}_{\lambda_0} \in \mathcal{D}$ and computing the residue $R\underline{s}$ [20]:

$$\underline{s} = \langle \underline{s}, \underline{g}_{\lambda_0} \rangle \underline{g}_{\lambda_0} + R\underline{s}, \tag{7.7}$$

where $\langle \underline{s}, \underline{g}_{\lambda_0} \rangle$ is the inner product between $\underline{s}$ and $\underline{g}_{\lambda_0}$. Since $R\underline{s}$ is orthogonal to $\underline{g}_{\lambda_0}$ we can write [20]:

$$||\underline{s}||^2 = |\langle \underline{s}, \underline{g}_{\lambda_0} \rangle|^2 + ||R\underline{s}||^2.$$

To minimize $||R\underline{s}||^2$, we need to choose $\underline{g}_{\lambda_0} \in \mathcal{D}$, such that $|\langle \underline{s}, \underline{g}_{\lambda_0} \rangle|$ is maximum, which is computationally done, by finding an almost optimal vector $\underline{g}_{\lambda_0}$ that satisfies [20]:

$$|\langle \underline{s}, \underline{g}_{\lambda_0} \rangle| \geq \alpha \sup_{\gamma \in \Gamma} |\langle \underline{s}, \underline{g}_\gamma \rangle|,$$

where $\alpha \in [0, 1]$ is an optimality factor [20]. By setting $R^0\underline{s} = \underline{s}$ and subdecomposing the residue at each iteration with the update rule for the mth iteration:

$$\text{choose } \underline{g}_{\lambda_{\gamma_m}} \text{ s.t. } |\langle R^m\underline{s}, \underline{g}_{\gamma_m} \rangle| \geq \alpha \sup_{\gamma \in \Gamma} |\langle R^m\underline{s}, \underline{g}_\gamma \rangle|, \tag{7.8}$$

we obtain the following decomposition of the mth residue at iteration m:

$$R^m\underline{s} = \langle R^m\underline{s}, \underline{g}_{\lambda_m} \rangle \underline{g}_{\lambda_m} + R^{m+1}\underline{s}. \tag{7.9}$$

If we stop after M terms, we obtain a sparse approximation of $\underline{s}$:

$$\hat{\underline{s}}(x) = \sum_{m=1}^{M} \langle \underline{s}, g_{\lambda_m} \rangle g_{\lambda_m}(x). \tag{7.10}$$

$$\tag{7.11}$$

As $M \to \infty$, this decomposition not only becomes exact, but also provides a sparse approximate representation if we set $M \ll \infty$. The adaptivity of the dictionary and the subsequent selection of the best fitting waveforms concentrate the energy of the data in as few possible waveforms, yielding a high non-linear approximation rate. However, this method often fails to locate two closely-spaced events in the coefficient vector. Due to the greediness of the algorithm, MP can also select the wrong basis function at the beginning (because two events are too close) and then spend the rest of the time correcting for its mistake [2].

7.4 Basis Pursuit

7.4.1 The formulation

Basis Pursuit (BP) is a method that decomposes a signal $\underline{s}$ into an optimal superposition of dictionary elements, where *optimal* means having the smallest ℓ^1 norm among all such decompositions [2]. BP requires that the decomposition of the signal $\underline{s}$ in the dictionary $\mathcal{D}$ be sparse, saying that we only need a few waveforms from $\mathcal{D}$ to build $\underline{s}$. Basis Pursuit (BP) solves the following minimization problem:

$$\min_{\underline{c}} ||\underline{c}||_1 \text{ , subject to } \underline{s} = \mathbf{\Phi}\underline{c}, \tag{7.12}$$

where $\mathbf{\Phi}$ is the matrix of the dictionary $\mathcal{D}$. BP tries to find the sparsest solution $\underline{c}$ by replacing the NP-hard ℓ^0 minimization by the ℓ^1 minimization stated above. The constraint imposed on the coefficient vector $\underline{c}$, requiring a minimization of the number of components, is very well suited to our type of problem.

7.4.2 Linear Programming (LP)

BP is a convex and non quadratic optimization problem that can be rewritten as a Linear Programming (LP) problem [2]:

$$\min_{\underline{x}} \underline{d}^T \underline{x} \text{ subject to } \boldsymbol{A}\underline{x} = \underline{b} \ , \ \underline{x} \geq 0, \tag{7.13}$$

where $\underline{x} \in \mathbb{R}^p$, $\underline{s} \in \mathbb{R}^n$, and $\underline{x} \geq 0$ requires all components of $\underline{x}$ to be positive. The reformulation of BP into the LP problem comes from the following change of variables [2]

$$p = 2m \ ; \ \underline{x} = (\underline{u}, \underline{v}) \ ; \ \underline{d} = (1,1) \ ; \ \boldsymbol{A} = (\boldsymbol{\Phi}, -\boldsymbol{\Phi}) \ ; \ \underline{b} = \underline{s}, \tag{7.14}$$

where $\underline{u}$ and $\underline{v}$ refer to the positive and negative components of $\underline{c}$ respectively ($\underline{u}, \underline{v} \in \mathbb{R}^m$), and $\underline{d}$ is a vector of ones in $\mathbb{R}^p$. First, we can write:

$$\begin{aligned} A\underline{x} &= (\boldsymbol{\Phi}, -\boldsymbol{\Phi})(\underline{u}, \underline{v}) && (7.15)\\ &= \boldsymbol{\Phi}(\underline{u} - \underline{v}) && (7.16)\\ &= \boldsymbol{\Phi}\underline{c} && (7.17) \end{aligned}$$

As $\boldsymbol{\Phi}\underline{c} = \underline{s} = \underline{b}$, (7.17) is equivalent to $A\underline{x} = \underline{b}$.

Secondly, we can write:

$$\begin{aligned} d^T\underline{x} &= (1,1)^T(\underline{u}, \underline{v}) && (7.18)\\ &= \underline{u} + \underline{v} && (7.19)\\ &= \sum_i u_i + \sum_j v_j && (7.20)\\ &= ||\underline{c}||_1 && (7.21) \end{aligned}$$

To clarify the notations stated above, consider the following example:

$$\underline{c} = \begin{bmatrix} 1 \\ -2 \\ 3 \\ -4 \end{bmatrix},$$

with $\underline{u} = \begin{bmatrix} 1 \\ 0 \\ 3 \\ 0 \end{bmatrix}$, $\underline{v} = \begin{bmatrix} 0 \\ 2 \\ 0 \\ 4 \end{bmatrix}$ and $\underline{c} = \underline{u} - \underline{v}$.

We can easily check that $\underline{d}^T \underline{x} = ||\underline{c}||_1$:

$$\begin{aligned} \underline{d}^T \underline{x} &= (1,1)^T (\underline{u}, \underline{v}) && (7.22) \\ &= \underline{u} + \underline{v} && (7.23) \\ &= \sum_i u_i + \sum_i v_i && (7.24) \\ &= 1 + 2 + 3 + 4 && (7.25) \\ &= ||\underline{c}||_1 && (7.26) \end{aligned}$$

The solution given by BP is obtained by an Interior-Point scheme [2] based on a Primal-Dual Log-Barrier method [2, 8]. The algorithm starts by an initial feasible solution yielded by the Method Of Frames [3], and proceeds by applying the Log-Barrier algorithm. The speed of the algorithm depends on the data, and the size and implementation of the dictionary, whereas the accuracy depends on the Conjugate Gradient solver.

7.4.3 Basis Pursuit denoising

Like any real data, seismic signals contain noise. If we decompose the entire noisy data $\underline{d} = \underline{s} + \underline{n}$ with respect to a dictionary $\mathcal{D}$, we also fit the noise. In order to avoid fitting the noise as well as the signal, we need to denoise the data and decompose the denoised data into the dictionary $\mathcal{D}$. In this case, we do not want an exact decomposition of $\underline{d}$, but an approximate one. Basis Pursuit denoising (BPDN) [2] adapts the original Basis Pursuit to the case of noisy data of the form $\underline{d} = \underline{s} + \underline{n}$, where $\underline{s}$ is the signal and $\underline{n}$ the noise. The denoising problem can be written as:

$$\min_{\underline{c}} (\frac{1}{2} ||\underline{d} - \mathbf{\Phi} \underline{c}||_2^2 + \lambda ||\underline{c}||_1), \tag{7.27}$$

where we minimize the ℓ^1 norm of the representation of the signal $\underline{s}$ that should be sparse in

the dictionary $\mathcal{D}$, as well as the data misfit. BPDN can be rewritten as a perturbed LP ([2, 8]), and is solved using a Simplex and an Interior Point method.

7.5 Basis Pursuit versus Matching Pursuit

7.5.1 Results

In this section, I present a comparison between Basis Pursuit and Matching Pursuit with respect to decomposing a signal $\underline{s}$ in a particular dictionary $\mathcal{D}$. The aim of the numerical experiments I made was to see whether the method involving the decomposition of the signal $\underline{s}$ using redundant dictionaries could find the right decomposition $\underline{c}$ and hence accurately localize the right waveforms in $\underline{s}$. For each dictionary $\mathcal{D}$, I started with a known coefficient vector $\underline{c}$, synthesized a signal $\underline{x} = \mathbf{\Phi}\underline{c}$, where $\mathbf{\Phi}$ was the matrix of $\mathcal{D}$ containing the different fractional-orders α, and then decomposed $\underline{x}$ using BP or MP. In my experiments I worked with the following parameters:

- The size of the dictionaries (number of different fractional-orders α)
- The number of nonzero components in the representation $\underline{c}$ (sparsity of $\underline{c}$)
- The distance between two different fractional-orders α

In all my experiments, I considered a signal $\underline{x}$ of length $n = 128$. In this section, I only present the results for the decimated fractional spline wavelet dictionary, abbreviated MDWT, as the results are the same for the fractional spline and undecimated fractional spline dictionaries.

Decimated Fractional Spline Wavelets

In the eight following figures, the top two plots show the superposition of the original signal $\underline{s}$, and the recovered signal. The top left figure shows the result by BP and the top right figure the result by MP. The two middle plots show the coefficient $\underline{c}$ found by BP (left) and MP (right). The two bottom plots show the original coefficient $\underline{c}$.

1. Coefficient $\underline{c}$ with only four nonzero components

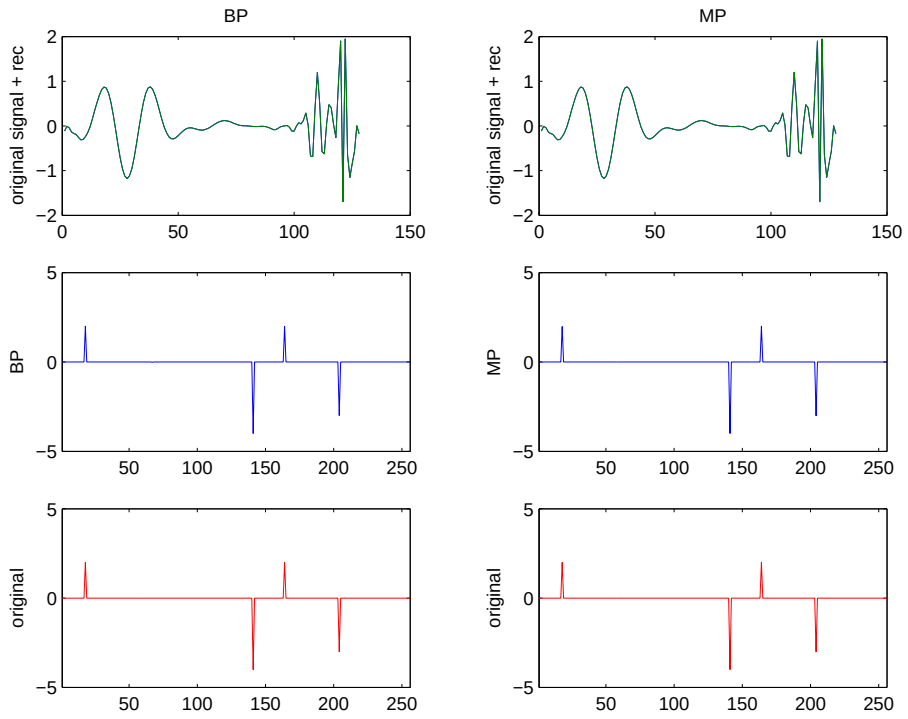

Figure 7.2: Analysis of a very sparse coefficient with an MDWT dictionary with 2 α

- Comparison between two dictionaries (small with two α and large with ten α)

  Figure 7.2 compares the efficiency of BP with the efficiency of MP for a dictionary containing only two different α ($\alpha_1 = 0$ and $\alpha_2 = 5$). The difference in α in $\mathcal{D}$ assures a certain dissimilarity between the atoms, and hence prevents $\mathcal{D}$ from being too badly conditioned. As we see in Figure 7.2, both BP and MP recover the right coefficient $\underline{c}$, first because there are very few nonzero entries in the original $\underline{c}$, and second because the dictionary $\mathcal{D}$ is small.

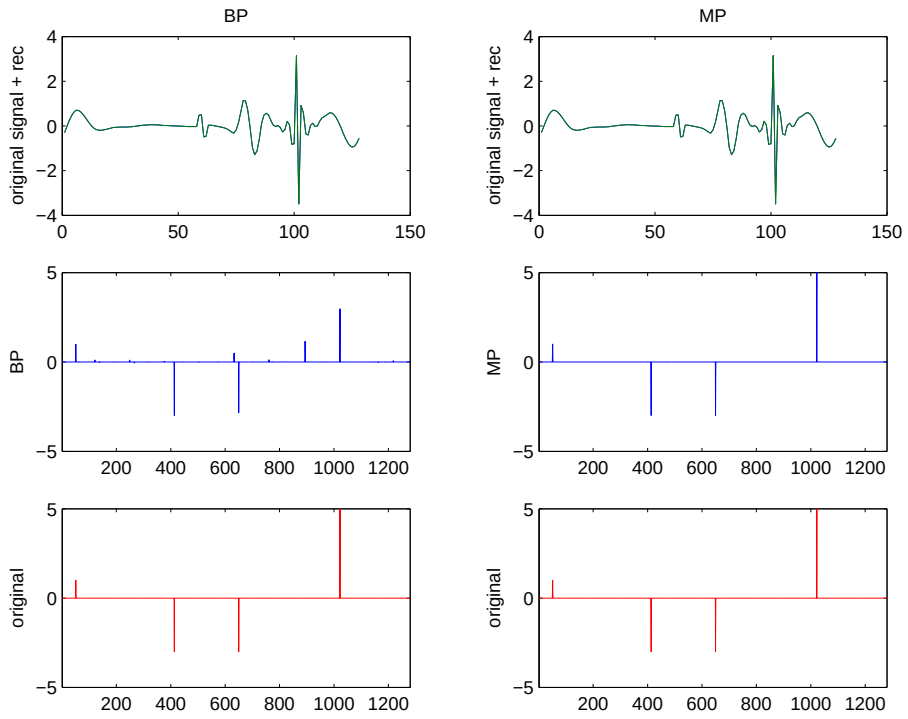

Figure 7.3: Analysis of a very sparse coefficient with an MDWT dictionary with 10 α

Figure 7.3 compares the efficiency of BP and MP for a larger dictionary $\mathcal{D}$ containing ten different α between 0 and 5. Because the different fractional-orders are closer to each other than in the previous example, the atoms in $\mathcal{D}$ are more similar. As we see in Figure 7.3, BP doesn't recover exactly the right coefficient $\underline{c}$, whereas MP does. The reason why BP doesn't, is because $\mathcal{D}$ is now much larger and $\underline{c}$ not sparse enough for $\mathcal{D}$. In fact, for this particular example, BP finds exactly the right coefficient $\underline{c}$ with a nonzero probability less than 1 (empirically found around 0.75) and we need less than four nonzero entries in $\underline{c}$ in order that BP finds exactly the right coefficient with probability 1.

- Comparison between two dictionaries (one with two very different α and one with two similar α)

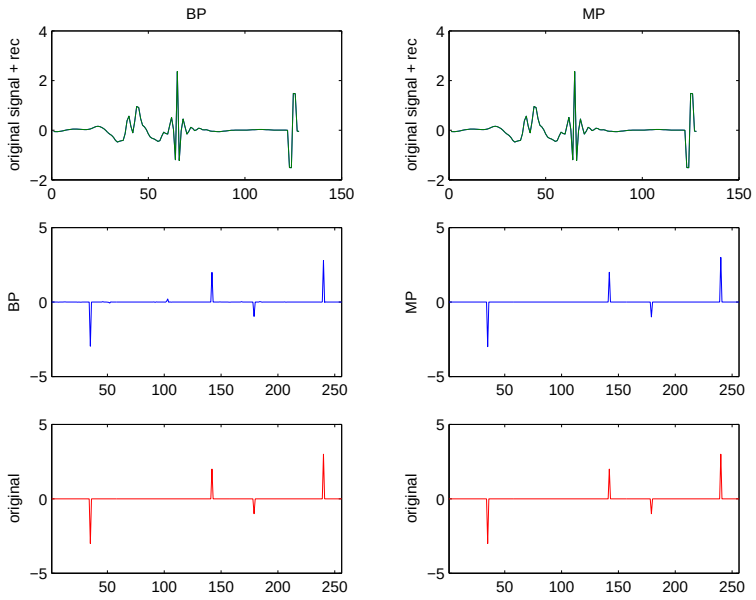

Figure 7.4: Analysis of a very sparse coefficient with an MDWT dictionary with 2 very different α

In Figure 7.4, the original coefficient vector $\underline{c}$ has only four nonzero entries, and both MP and BP find it exactly. In this case, the dictionary $\mathcal{D}$ is very small and contains dissimilar atoms. For this specific dictionary, $\underline{c}$ is sparse enough so BP and MP can find it.

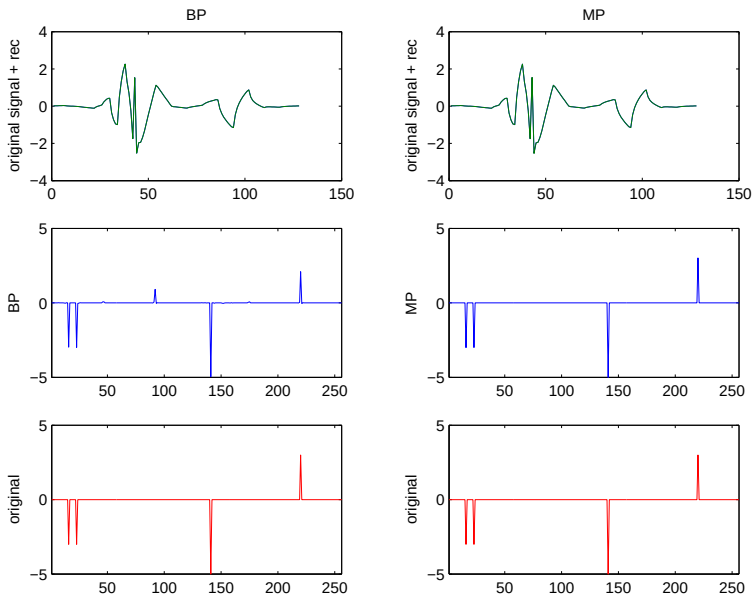

Figure 7.5: Analysis of a very sparse coefficient with an MDWT dictionary with 2 similar α

In Figure 7.5 however, only MP finds the original coefficient exactly. BP finds the nonzero entries of the original $\underline{c}$, but adds another nonzero one. BP has more trouble finding the right coefficient because the atoms in $\mathcal{D}$ are much more similar. In this case, the original $\underline{c}$ is not sparse enough for BP to find it with probability 1. Like in Figure 7.3, BP finds the right coefficient $\underline{c}$ exactly with a nonzero probability less than 1 (empirically found around 0.75). The more similar the α, the lower the probability that BP finds the right coefficient $\underline{c}$ for a given original $\underline{c}$ with a fixed number of nonzero entries. MP has the same problem, but seems to be able to recover coefficients that are less sparse than the coefficients BP can recover.

2. Coefficient $\underline{c}$ with ten nonzero components

- Comparison between two dictionaries (small with two α and large with ten α)

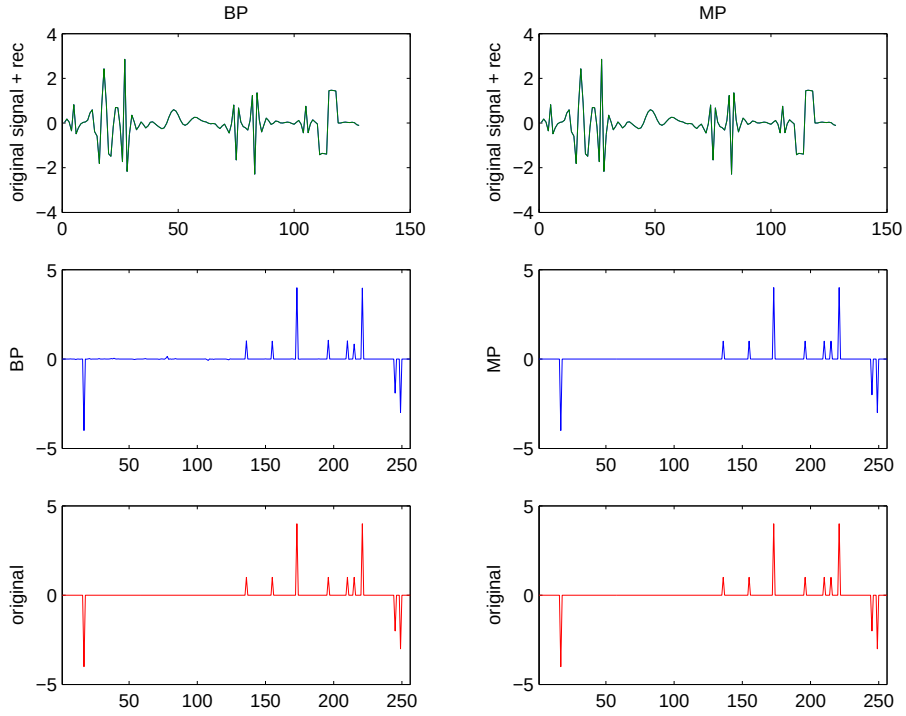

Figure 7.6: Analysis of a less sparse coefficient with an MDWT dictionary with 2 α

Figure 7.6 shows the efficiency of BP and MP in a small dictionary for a given original coefficient $\underline{c}$. In this case, both BP and MP find the original coefficient exactly. The original coefficient, although containing 10 nonzero entries, is sparse enough for MP and BP to find it in this dictionary $\mathcal{D}$.

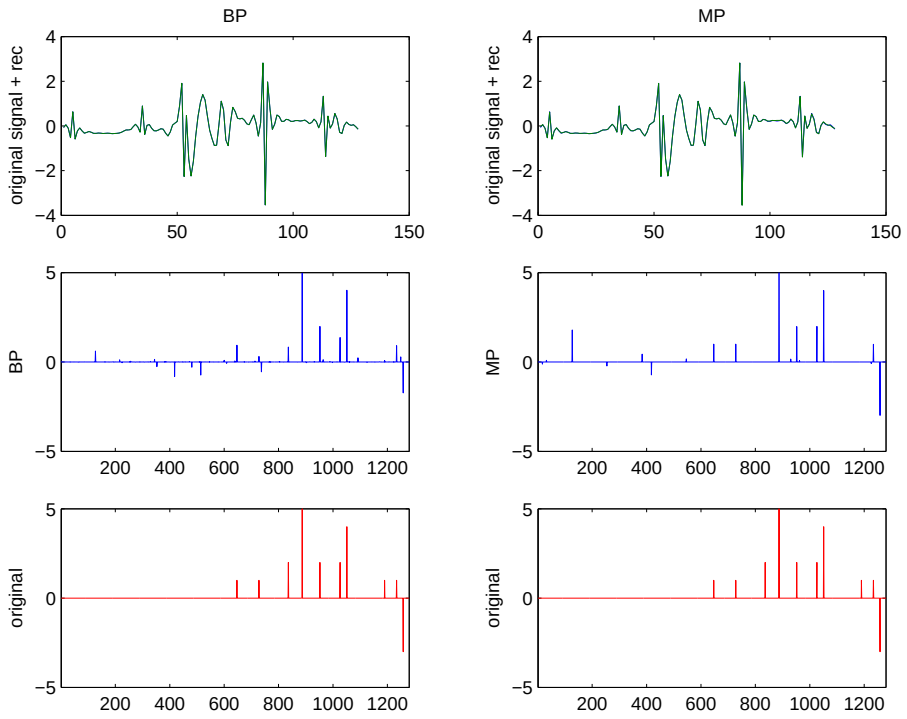

Figure 7.7: Analysis of a less sparse coefficient with an MDWT dictionary with 10 α

Unlike in Figure 7.6, Figure 7.7 shows that neither BP nor MP exactly finds the right coefficient back, although MP seems a little bit better. In this case, the dictionary is too large for BP and MP to find the original coefficient back. For this particular dictionary with ten different α, the original $\underline{c}$ is not sparse enough for BP and MP to find it exactly. The reason for this result is explained by the Spark of the dictionary: the larger $\mathcal{D}$, the smaller its Spark, and hence the sparser $\underline{c}$ needs to be in order to be found by BP and MP. This condition is given in the equivalence theorem mentioned in section (3.2) of chapter 3, and the definition of the Spark is given in section **??** of chapter 3.

- Comparison between two dictionaries (one with two very different α and one with two similar ones)

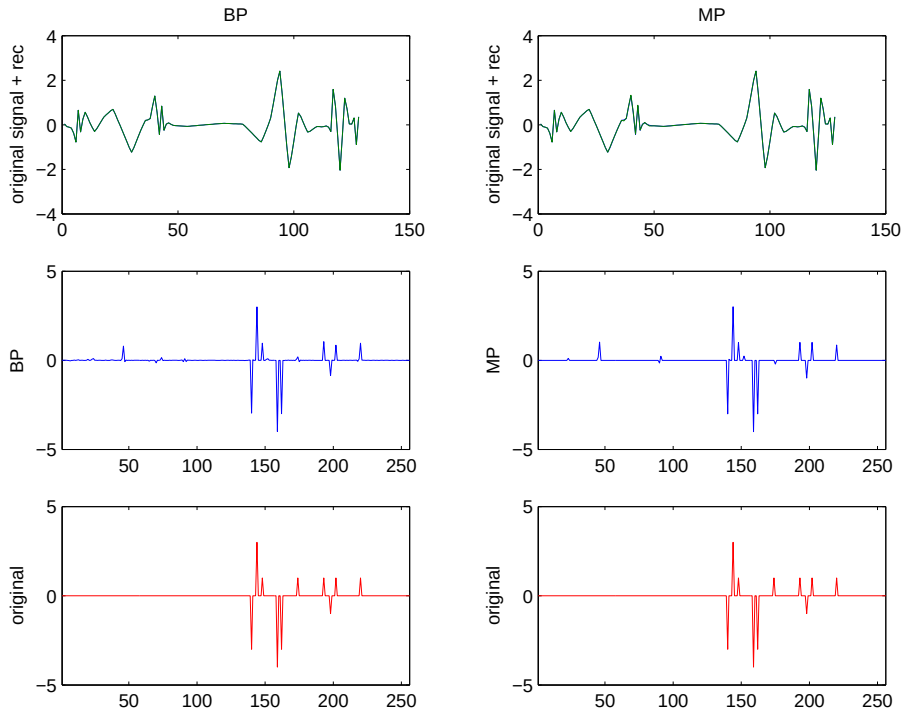

Figure 7.8: Analysis of a less sparse coefficient with an MDWT dictionary with 2 close α

In Figure 7.8, $\underline{s}$ is decomposed in a small dictionary $\mathcal{D}$ with two very similar α (0.5 and 1). When the α are similar however, neither BP nor MP recover exactly the right coefficient. In fact, both recover the initial coefficient with a nonzero probability strictly less than 1. The original coefficient $\underline{c}$ is not sparse enough.

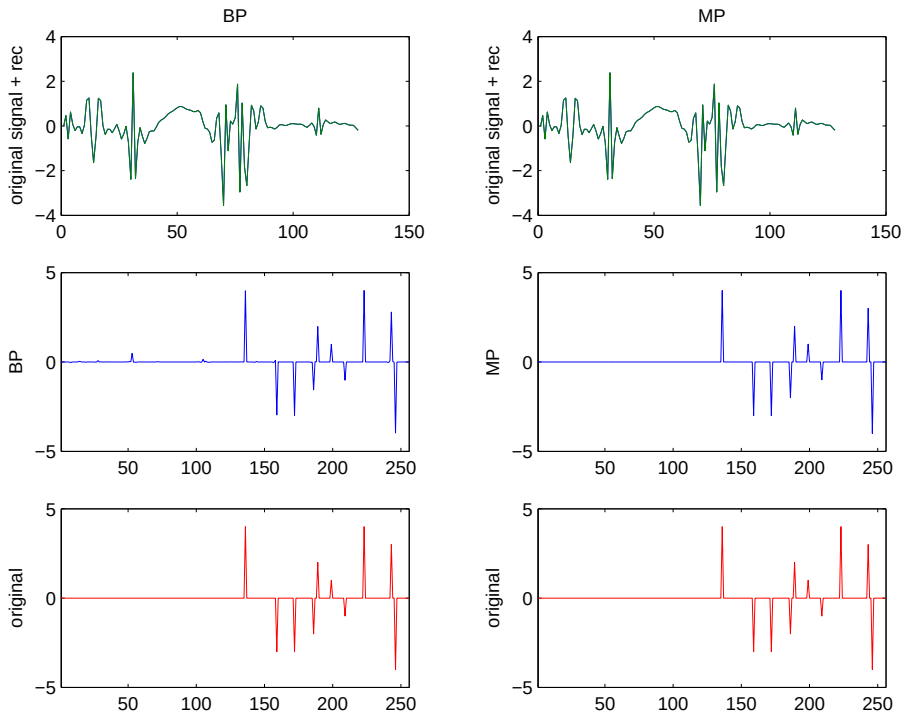

Figure 7.9: Analysis of a less sparse coefficient with an MDWT dictionary with 2 different α

In Figure 7.9, $\underline{s}$ is decomposed in a small dictionary $\mathcal{D}$ with two very different α (0 and 5). When the α are very different, BP and MP can recover the right coefficient, BP doesn't recover it exactly but is very close to the right coefficient.

3. Coefficients $\underline{c}$ with two very close components

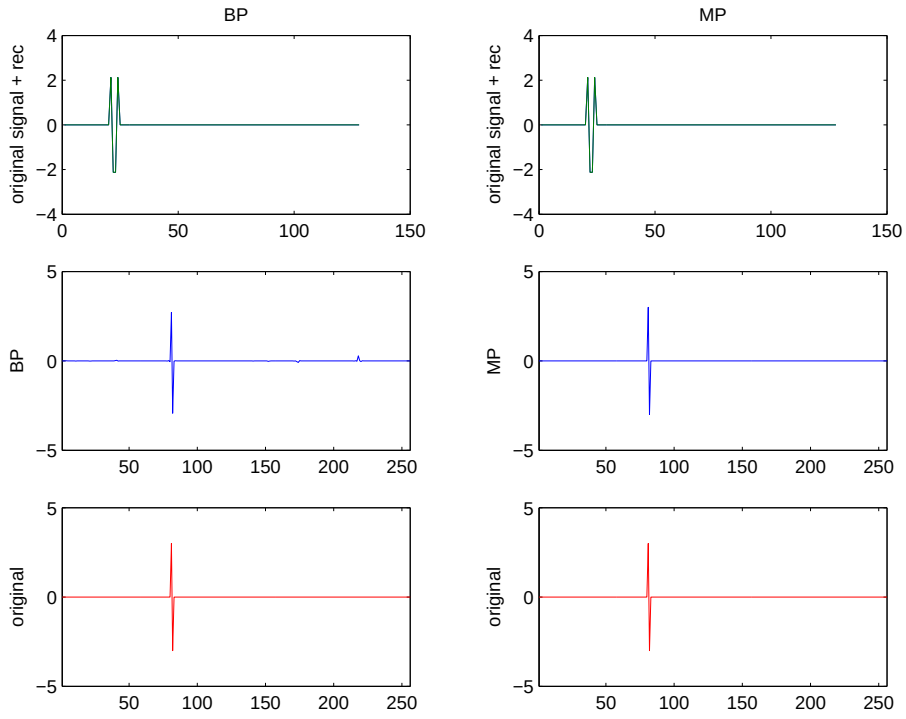

Figure 7.10: Analysis of two close components in an MDWT dictionary with two different α

Figures 7.10 and 7.11 compare the efficiency of BP and MP in resolving two close components in $\underline{c}$. In Figure 7.10, the dictionary $\mathcal{D}$ is small and contains only two different α. In this case, both BP and MP recover exactly the right coefficient $\underline{c}$, no matter how close the components are.

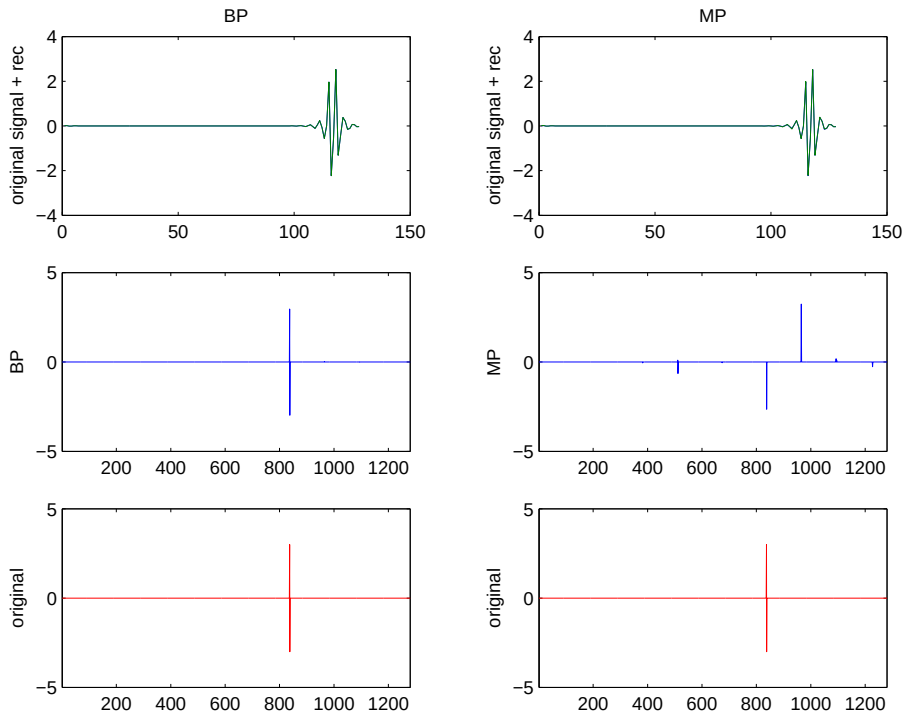

Figure 7.11: Analysis of two close components in an MDWT dictionary with ten different α

In Figure 7.11, $\mathcal{D}$ is large and contains ten different α between 0 and 5. In this case, BP finds exactly the right coefficient (multiresolution), but MP cannot resolve the two close components. In small enough dictionaries MP can resolve two close components, but fails to do so in larger ones. BP is more sensitive to the number of nonzero entries than to how close the events in the original coefficient $\underline{c}$ are.

7.5.2 Discussion

I showed some results for dictionaries made with decimated fractional spline wavelets and got similar results for fractional spline and undecimated fractional spline wavelet dictionaries. The results revealed the importance of the size of the dictionary (its redundancy), the orders α and the number of nonzero entries in the coefficient $\underline{c}$. There is a trade off between the redundancy of the dictionary $\mathcal{D}$ and the sparsity of $\underline{c}$. For a given dictionary $\mathcal{D}$, there is a maximum number of nonzero components in $\underline{c}$, that guarantees its being the unique solution of (P_1) (and hence (P_0)). The definitions of the (P_0) and (P_1) problems are given in section **??** of chapter 3. For our type of dictionaries (fractional B splines, and decimated and undecimated fractional spline wavelets), the results are as follows:

1. If $\mathcal{D}$ is small (e.g. made of only 2 different α) and the α very different, the maximum number of nonzero entries in $\underline{c}$ is around 10.

2. If $\mathcal{D}$ is small (e.g. made of only 2 different α) and the α more similar, the maximum number of nonzero entries in $\underline{c}$ is close to 10. BP and MP find the correct coefficient back with a high probability.

3. If $\mathcal{D}$ is large (e.g. made of 10 different α or more), the maximum number of nonzero entries in $\underline{c}$ is very small (around 1 or 2 depending on $\mathcal{D}$).

From our results, we can infer two major problems:

1. Seismic data is made of two many different events for our type of dictionaries.

2. Seismic data contains a wide variety of fractional-orders, which requires large dictionaries.

We therefore cannot rely on BP or MP to decompose and find the fractional-orders α of a seismic trace. Another way of tackling the problem is to consider $\underline{s}$ as a sum of different signals $\underline{s}_i$, each having a different fractional-order α_i. Using the Block Coordinate Relaxation Method (BCR) which is an iterative algorithm, we hope to be able to separate the different signals.

7.6 Block Coordinate Relaxation Method

7.6.1 Problem statement

We consider the seismic signal $\underline{s}$ as:

$$\underline{s} = \underline{s}_{\alpha_1} + ... + \underline{s}_{\alpha_n} + \underline{r},$$

where $\underline{s}_{\alpha_i}$ is an α_i-th order signal, and $\underline{r}$ is the residual containing the rest of the orders of $\underline{s}$ that we didn't include in our model ($\underline{s}_{\alpha_1} + ... + \underline{s}_{\alpha_n}$). $\underline{r}$ can also contain the noise if we work with noisy data. Given $\underline{s}$, we would like to find its different components $\underline{s}_{\alpha_i}$, and in particular its associated orders α_i. Like BP, we will try to find the representation $\underline{c}$ of the signal $\underline{s}$, with respect to an overcomplete dictionary $\mathcal{D}$ made of waveforms parameterized by their fractional-orders α. Examples of such dictionaries include fractional B spline, and decimated and undecimated fractional spline wavelet dictionaries.

We consider a dictionary $\mathcal{D}$ as

$$\mathcal{D} = [\mathcal{D}_{\alpha_1}, ..., \mathcal{D}_{\alpha_n}],$$

and its matrix $\mathbf{\Phi}$:

$$\mathbf{\Phi} = [\mathbf{\Phi}_{\alpha_1}, ..., \mathbf{\Phi}_{\alpha_n}],$$

where $\mathbf{\Phi}_{\alpha_i}$ is a the matrix of $\mathcal{D}_{\alpha_i}$ containing the different translations of α_i-th order waveforms. A representation $\underline{c}$ in $\mathcal{D}$ is of the form $\underline{c} = [\underline{c}_{\alpha_1}, ..., \underline{c}_{\alpha_n}]$. Following these notations, we can write our model as

$$\underline{s} = \mathbf{\Phi}\underline{c} + \underline{r}. \tag{7.28}$$

In the rest of the paper, we will use the following notations:

$$\underline{s}_{\alpha_i} = \underline{s}_i \ , \quad \underline{c}_{\alpha_i} = \underline{c}_i \quad \text{and} \quad \mathbf{\Phi}_{\alpha_i} = \mathbf{\Phi}_i$$

Our main assumption here, is that the representation of each $\underline{s}_i$ is sparse in the dictionary $\mathcal{D}_i$, so that we only need a few atoms in $\mathcal{D}_i$ to build $\underline{s}_i$. This is the same idea as BP, because we

would like each α_i-th order event of the signal $\underline{s}$ to be described with as few atoms as possible: ideally, each event would be described by only one atom and hence the order α_i of this event would be read on its associated atom.

We propose to solve the following minimization problem:

$$\min_{\underline{c}_1,\ldots,\underline{c}_n} (\underbrace{\beta_1||\underline{c}_1||_1 + \ldots + \beta_n||\underline{c}_n||_1}_{Part1} + \underbrace{||\underline{s} - \mathbf{\Phi}\underline{c}||_2^2}_{Part2}), \tag{7.29}$$

where $\beta_i > 0$, for $0 \leq i \leq n$. The parameters β_i are trade-off parameters between the different ℓ^1 norms and the data misfit.

1. The first part of the functional minimizes the representation of each signal $\underline{s}_i$ in the corresponding dictionary $\mathcal{D}_i$.

2. The second part minimizes the data misfit. This second part relaxes the constraint. If an additional content existing in the signal $\underline{s}$ is not represented sparsely by the dictionaries $\mathcal{D}_i$, the above formulation will tend to allocate this content in the residual [25].

Remark: What makes the minimization (7.29) succeed in separating the different signals $\underline{s}_i$, is that each dictionary $\mathcal{D}_i$ is very good in representing $\underline{s}_i$ and bad in representing the other $\underline{s}_j$, for $j \neq i$. The better this assumption is verified, the more accurate the algorithm.

7.6.2 Algorithm

I solved the minimization (7.29) using a slightly modified version of the Block Coordinate Relaxation Method (BCR) implemented by Starck et al. [25]. If $\mathbf{\Phi}_i^* \mathbf{\Phi}_i = I$, the previous minimization can be solved by Soft Thresholding [25].

If we fix every coefficient vector $\underline{c}_i$ except $\underline{c}_{i_0}$, we can solve for $\underline{c}_{i_0}$ with:

$$\min_{\underline{c}_{i_0}} \beta_{i_0}||\mathbf{\Phi}_{i_0}^* \underline{c}_{i_0}||_1 + ||\underline{S}_{i_0} - \mathbf{\Phi}_{i_0}\underline{c}_{i_0}||_2^2 \tag{7.30}$$

where $\underline{S}_{i_0} = \underline{s} - \sum_{i=1, i\neq i_0}^{n} \underline{s}_i$. If $\mathbf{\Phi}_i^* \mathbf{\Phi}_i = I$, (7.30) is equivalent to:

$$\min_{\underline{c}_{i_0}} \beta_{i_0}||\underline{c}_{i_0}||_1 + ||\mathbf{\Phi}_{i_0}^* \underline{S}_{i_0} - \underline{c}_{i_0}||_2^2, \tag{7.31}$$

which can be written as:

$$\sum_{k=1}^{N}(|\underline{c}_{i_0}(k)| + \lambda(c_{i_0}(k) - Z_{i_0}(k))^2),$$

where $Z_{i_0}(k)$ is the k-th component of $\mathbf{\Phi}^*_{i_0}\underline{S}_{i_0}$ and $\lambda = \frac{1}{\beta_{i_0}}$. This is equivalent to N scalar, convex and independent minimization problems that are solved by Soft Thresholding applied componentwise to $\underline{c}_{i_0}$ [4]. The minimization can still be solved by Soft Thresholding [24] when the matrices are a concatenation of matrices A_i that satisfy $A_i^* A_i = I$. See Appendix C for the derivation of the soft thresholding solution. The undecimated orthonormal wavelet transform consists of all shifted versions of the decimated orthonormal wavelet transform. The matrix of the decimated orthonormal wavelet transform is unitary, and the matrix of the undecimated wavelet transform is a concatenation (modulus a reordering) of unitary matrices. We can hence use Soft Thresholding to solve the minimization problem when using undecimated orthonormal wavelet transforms.

7.6.3 Numerical scheme

I present the basic numerical scheme for separating two signals $\underline{s}_1$ and $\underline{s}_2$, assuming $\underline{s}$ is of the form $\underline{s} = \underline{s}_1 + \underline{s}_2 + \underline{r}$, where $\underline{s}_1$ is an α_1-th order signal, $\underline{s}_2$ is an α_2-th order signal and $\underline{r}$ is the residual which can contain other α_i-th order signals and/or noise. The dictionary we use can be written as: $\mathbf{\Phi} = [\mathbf{\Phi}_1, \mathbf{\Phi}_2]$, where $\mathbf{\Phi}_1$ is adapted to $\underline{s}_1$, and $\mathbf{\Phi}_2$ to $\underline{s}_2$.

NUMERICAL SCHEME:

`Initialize Lmax, number of iterations, and thresholds` $\delta_1 = \beta_1.L_{max}$ and $\delta_2 = \beta_2.L_{max}$

Perform L_{max} times

`Part A: Update` $\underline{c}_2$ `assuming` $\underline{c}_1$ `is fixed`

$\underline{c}_2^{old} = \underline{c}_2$

- `Synthesize the signal` $\underline{s}_1 = \mathbf{\Phi}_1 \underline{c}_1$
- `Calculate the residual` $\underline{R}_2 = \underline{s} - \underline{s}_1$
- `Calculate` $\underline{c}_2 = \underline{c}_2^{old} + \mathbf{\Phi}_2^T(\underline{R}_2 - \mathbf{\Phi}_2 \underline{c}_2^{old})$
- `Soft threshold` $\underline{c}_2$ `with` δ_2, `i.e. solves the following minimization problem`:

$$\min_{\underline{c}_2} \delta_2 ||\underline{c}_2||_1 + ||\underline{R}_2 - \underline{s}_2||_2^2$$

`Update the threshold` $\delta_2 = \delta_2 - \beta_2$

`Part B: Update` $\underline{c}_1$ `assuming` $\underline{c}_2$ `is fixed`

$\underline{c}_1^{old} = \underline{c}_1$

- `Synthesize the signal` $\underline{s}_2 = \mathbf{\Phi}_2 \underline{c}_2$
- `Calculate the residual` $\underline{R}_1 = \underline{s} - \underline{s}_2$
- `Calculate` $\underline{c}_1 = \underline{c}_1^{old} + \mathbf{\Phi}_1^T(\underline{R}_1 - \mathbf{\Phi} \underline{c}_1^{old})$
- `Soft threshold` $\underline{c}_1$ `with` δ_1, `i.e. solves the following minimization problem`:

$$\min_{\underline{c}_1} \delta_1 ||\underline{c}_1||_1 + ||\underline{R}_1 - \underline{s}_1||_2^2$$

`Update the threshold` $\delta_1 = \delta_1 - \beta_1$

`end`

The above algorithm is given for only two signals, but I programmed a general algorithm for any number of signals and any dictionaries.

7.6.4 Simple examples

I applied the above algorithm to two examples: one that I made myself and one taken from Starck et al. [26]. In both examples we are dealing with data d of the form $d = f_1 + f_2 + n$, where f_1 and f_2 are two different signals, and n is the noise.

1. In the first example, f_1 is a cosine function and f_2 is a step function. I used the Discrete Cosine Transform as the dictionary $\mathbf{\Phi}_1$ adapted to f_1 and the Haar wavelet basis as the dictionary $\mathbf{\Phi}_2$ adapted to f_2. The results are shown in Figure 7.12.

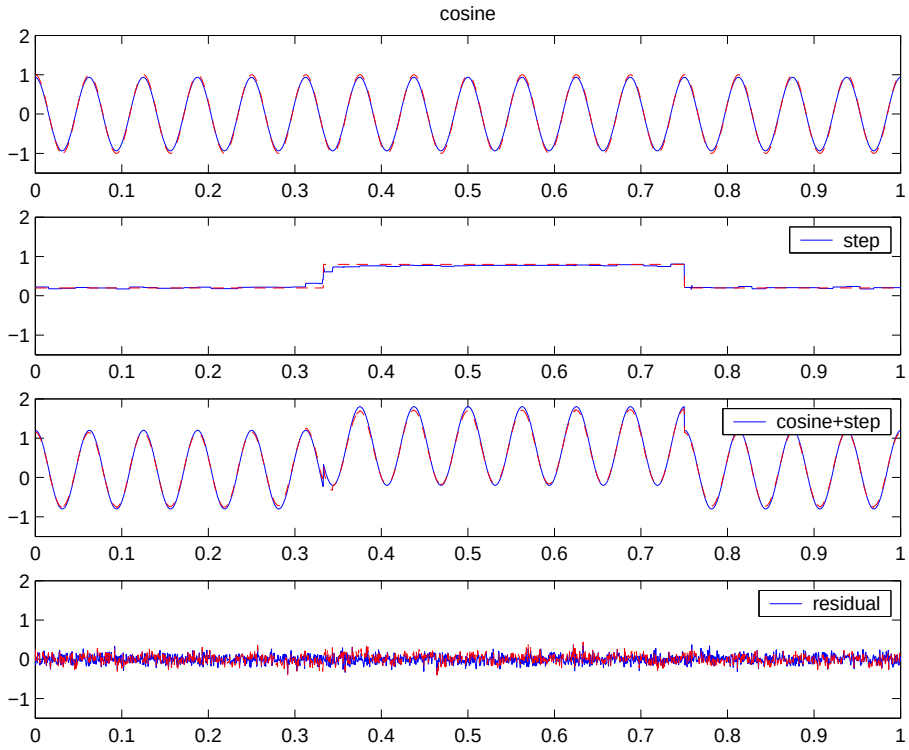

Figure 7.12: BCR algorithm: separating the noise, a cosine function and a step function.

From top to bottom: original cosine function (blue line) and recovered cosine function (red line), original step function (blue line) and recovered step function (red line), sum of the original signals (blue line), sum of the recovered signals (red line) and residual.

2. In Starck's example [25], f_1 is a cosine function and f_2 consists of bumps (Gaussians). I used the Discrete Cosine Transform as the dictionary $\mathbf{\Phi}_1$ and the ATrou wavelet basis as the dictionary $\mathbf{\Phi}_2$. The results are shown in Figure 7.13.

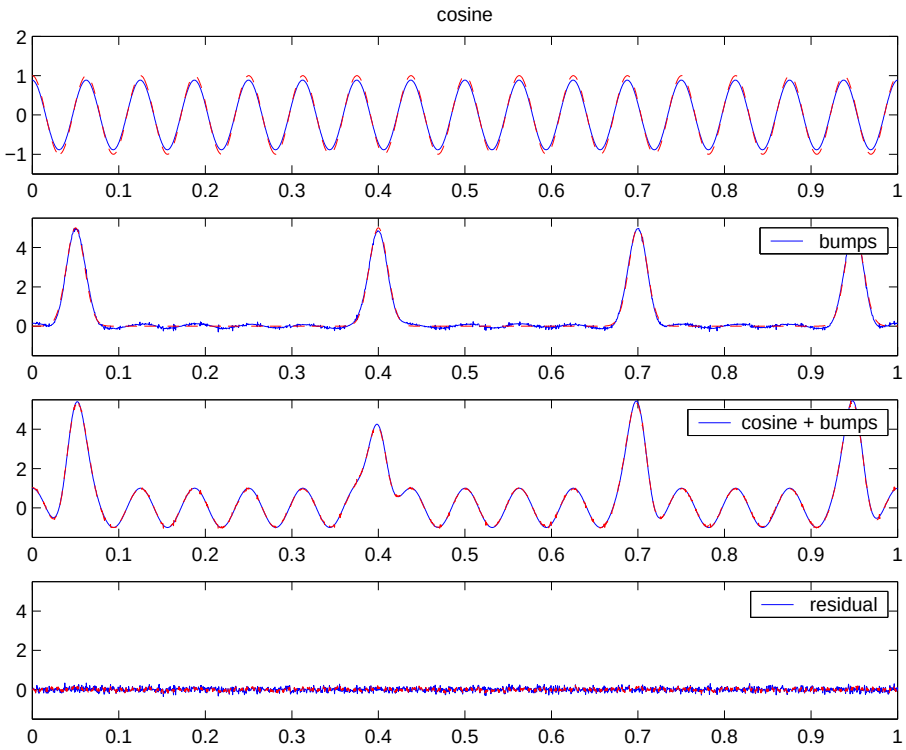

Figure 7.13: BCR algorithm: separating the noise, a cosine function and bumps.

From top to bottom: original cosine function (blue line) and recovered cosine function (red line), original bumps (blue line) and recovered bumps (red line), sum of the original signals (blue line), sum of the recovered signals (red line) and residual.

7.6.5 Multiple Block Coordinate Relaxation Method

The multiple BCR iteratively performs a BCR decomposition on each of the separated signals resulting from a previous BCR decomposition. The idea is to run the BCR iteratively on the different signal contents, until we separate them quasi perfectly. I empirically noticed that running the BCR more than three times does not really improve the results. I limited the iterations to three.

7.6.6 Results

I present here some results on the performance of the BCR algorithm for the decimated fractional spline wavelet dictionary. The results are similar for fractional B splines and undecimated fractional spline wavelet dictionaries. I tested the BCR algorithm for different settings and different synthetic seismic signals that I generated randomly in order to avoid the creation of patterns. The experiments test the following cases:

1. The analyzing dictionary $\mathcal{D}$ exactly contains the waveforms in $\underline{s}$
2. The analyzing dictionary $\mathcal{D}$ contains the waveforms in $\underline{s}$ as well as other ones
3. The analyzing dictionary $\mathcal{D}$ doesn't contain the waveforms in $\underline{s}$

The synthetic signal $\underline{s}$ was built as a real seismic signal by randomly building its different coefficients. By hypotheses, $\underline{s}$ is a summation of signals $\underline{s}_i$ with different fractional-orders α_i. In order to construct $\underline{s}$, I created each coefficient $\underline{c}_i$ with random position and number of nonzero entries, and synthesized $\underline{s}_i$ by computing $\underline{s}_i = \mathbf{\Phi}_i \underline{c}_i$, where $\mathbf{\Phi}_i$ is the matrix of the dictionary $\mathcal{D}_i$ associated to the order α_i. Given $\underline{s}$, the BCR algorithm tries to separate the different content types of $\underline{s}$ in the analyzing dictionary $\mathcal{D}$.

1. **The analyzing dictionary $\mathcal{D}$ exactly contains the waveforms in $\underline{s}$**

   - Dictionary $\mathcal{D}$ made of 4 different α between 0 and 5

     In this example, $\underline{s}$ is synthesized and analyzed with the same dictionary $\mathcal{D}$ and the coefficient $\underline{c}$ of the signal $\underline{s}$ in $\mathcal{D}$ has 27 nonzero entries.

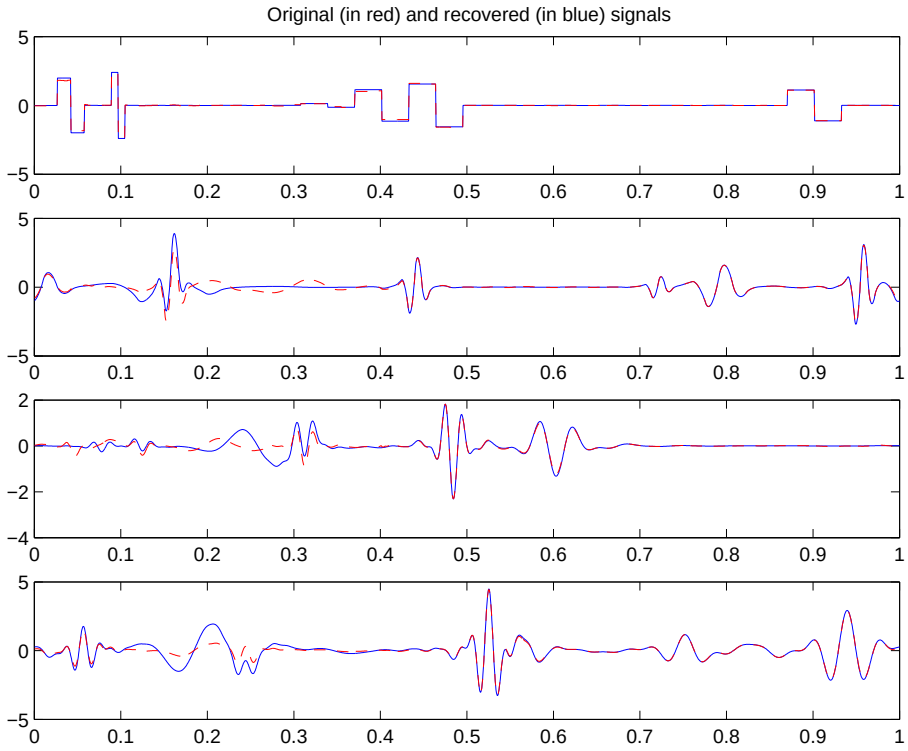

Figure 7.14: Original and recovered signals

Figures 7.14 shows that the different content types in $\underline{s}$ are successfully separated and the coefficients exactly recovered. With 27 nonzero entries in $\underline{c}$, the BCR algorithm is able to separate the different content types in $\underline{s}$ and finds the right coefficient back. This is an improvement over BP and MP.

Figure 7.15 shows the superposition of the original coefficients and the coefficients $\underline{c}_i$ found by the BCR algorithm. The original coefficient has 27 nonzero entries. The different coefficients to the different content types are perfectly recovered by the algorithm.

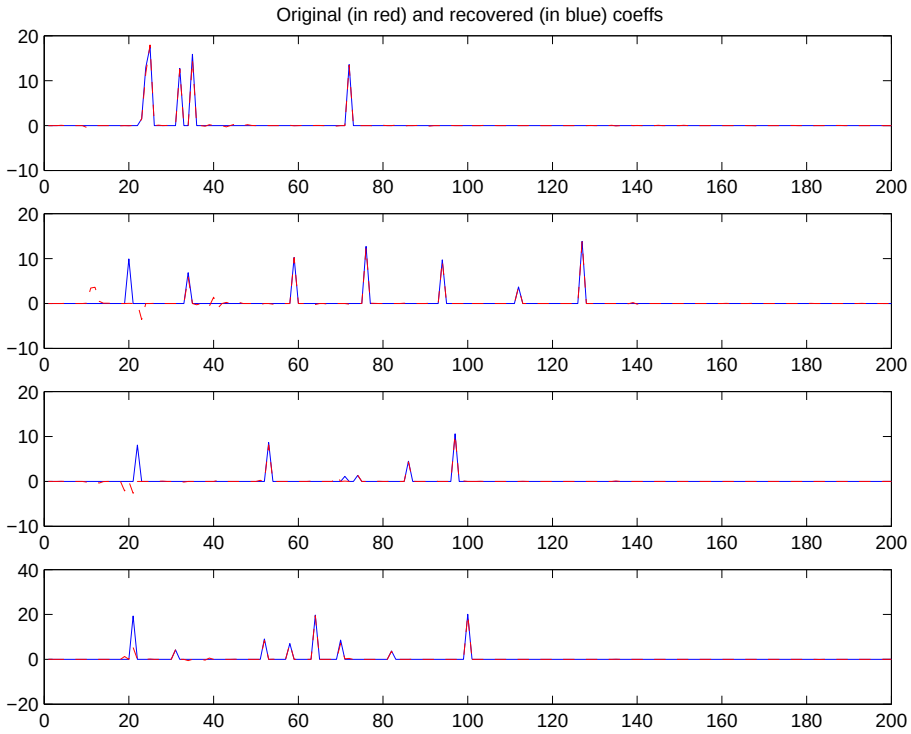

Figure 7.15: Original and recovered coefficients

- Dictionary made of 10 different α between 0 and 1

  The original coefficient $\underline{c}$ has 33 nonzero entries.

  In Figures 7.16 and 7.17, we can see that the separation of the different signals is not very successful. Because the dictionary is large and the waveforms very similar (the fractional-orders are much closer), the dictionaries become more similar. In particular, one dictionary $\mathbf{\Phi}_i$ is not bad in representing the other signals $\underline{s}_j$ for $j \neq i$, which is why the separation is not perfect. The more similar the dictionaries, the less the BCR algorithm will be able to separate the different content types of $\underline{s}$.

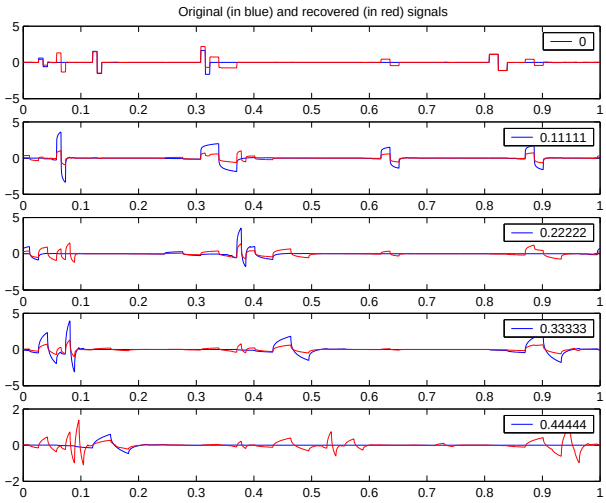

Figure 7.16: Five first original and recovered signals

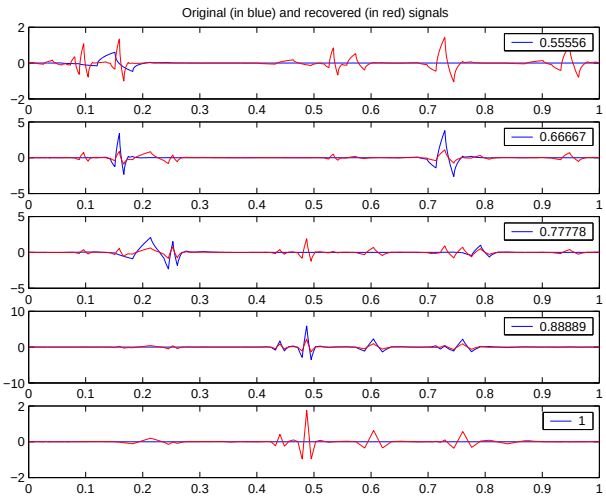

Figure 7.17: Five last original and recovered signals

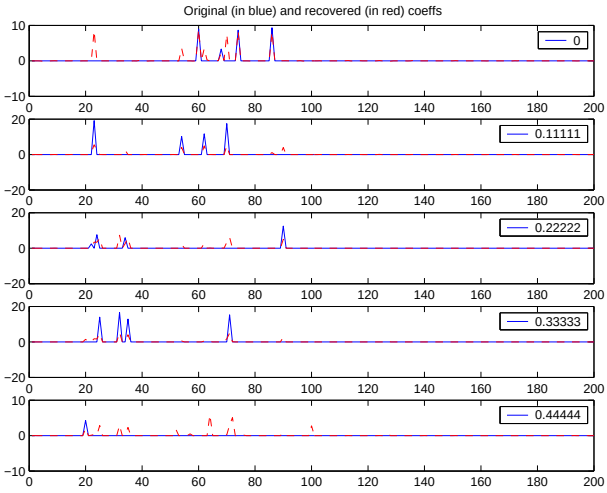

Figure 7.18: Five first original and recovered coefficients

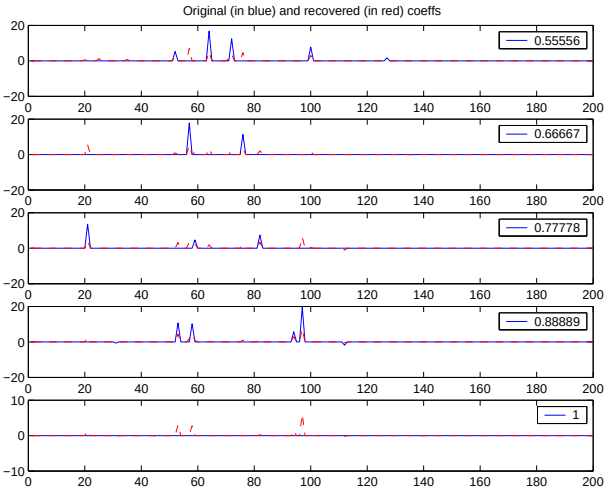

Figure 7.19: Five last original and recovered coefficients

Figures 7.18 and 7.19 illustrate the performance of the BCR algorithm on the coefficients.

2. **The analyzing dictionary $\mathcal{D}$ contains the waveforms in $\underline{s}$ as well as other ones**

- *Synthesizing dictionary made of 2 α between 0 and 1 and analyzing dictionary made of 4 α between 0 and 3*

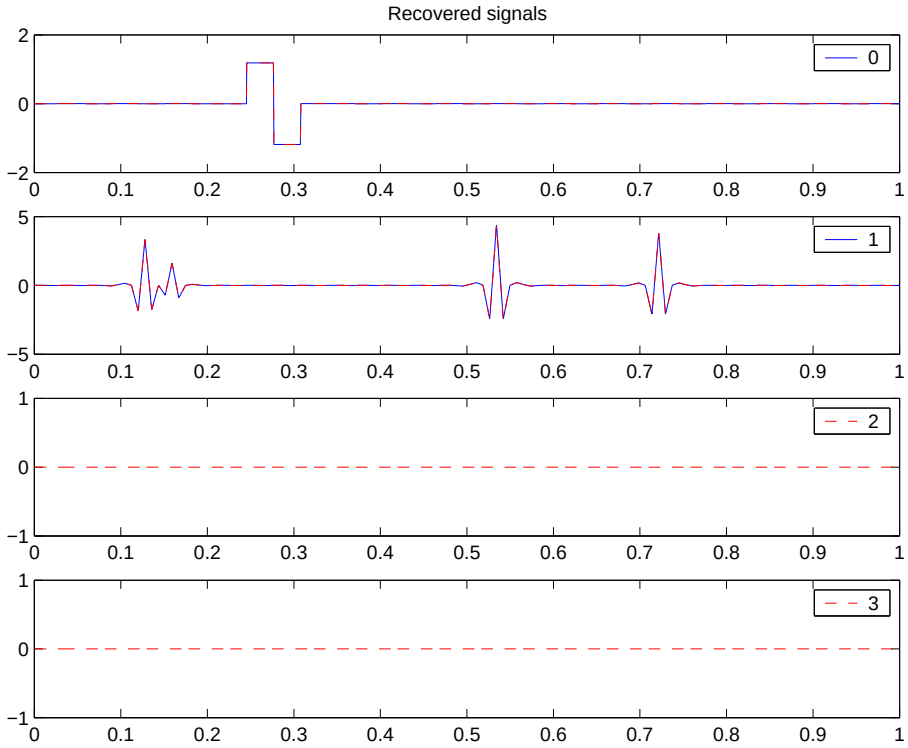

Figure 7.20: Original and recovered signals

In this example, the synthetic signal $\underline{s}$ is only made of two different content types: one with order 0 and one with order 1, and the original coefficient has only 5 nonzero entries. The analyzing dictionary is made of four different α (0,1,2 and 3). $\underline{s}$ is decomposed into 4 signals with order 0,1,2 and 3 respectively. Figure 7.20 shows the original signals in blue (top two plots) and the separated ones in dashed red (in the four plots). The separation is perfect.

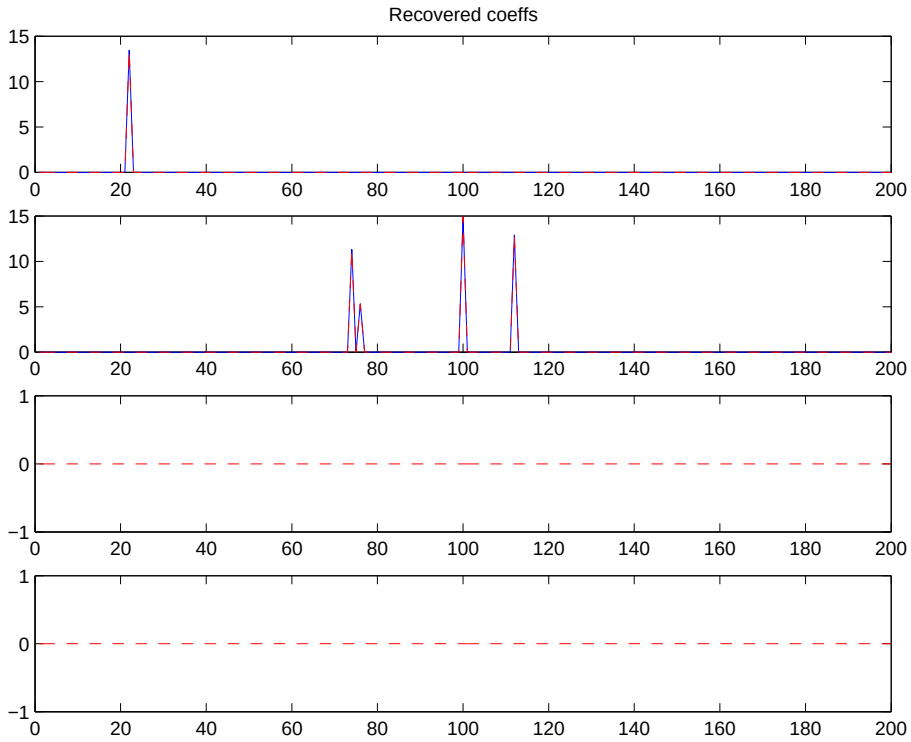

Figure 7.21: Original and recovered coefficients

Figure 7.21 shows the coefficients of the different signals in the same way as for the signals. They are perfectly recovered.

- *Synthesizing dictionary made of 2 α between 0 and 1 and analyzing dictionary made of 4 α between 0 and 1*

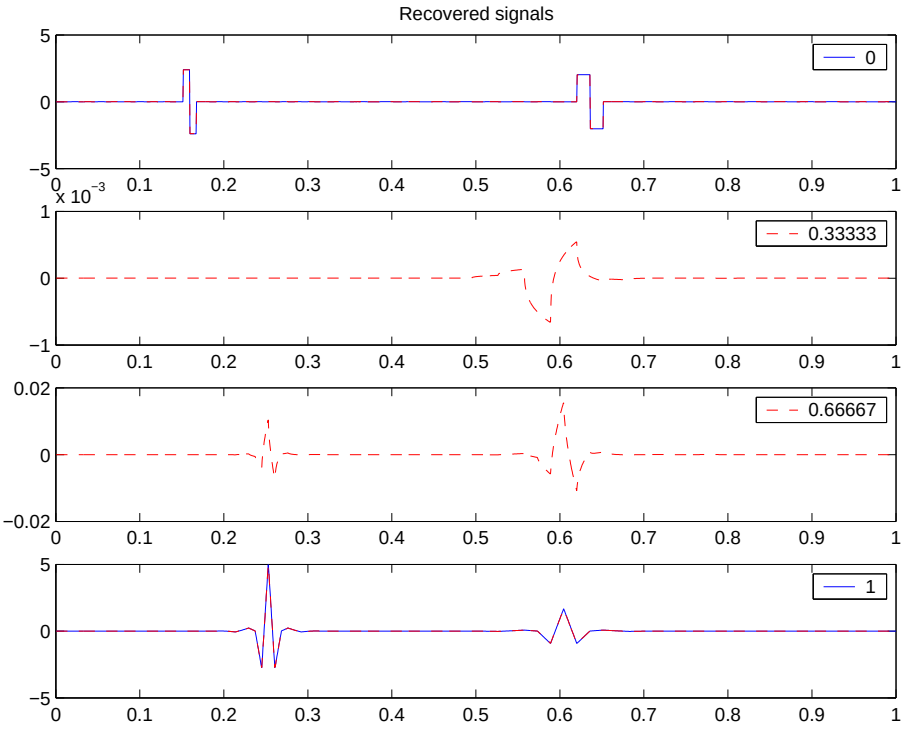

Figure 7.22: Original and recovered signals

In this case, a similar synthetic signal $\underline{s}$ with 4 events is decomposed with respect to a different analyzing dictionary. The analyzing dictionary contains four different fractional-orders α ranging from 0 and 1 ($0, \frac{1}{3}, \frac{2}{3}$ and 1). Figure 7.22 shows the original signals in blue (top and bottom plots) and the separated ones in dashed red (in the four plots). The separation is not perfect because the waveforms in $\mathcal{D}$ are too similar, however, it is still quite good as the amplitudes of the signals with orders $\frac{1}{3}$ and $\frac{2}{3}$ are very small. The non optimal choice of $\mathcal{D}$ for the BCR algorithm affects the performance of the separation.

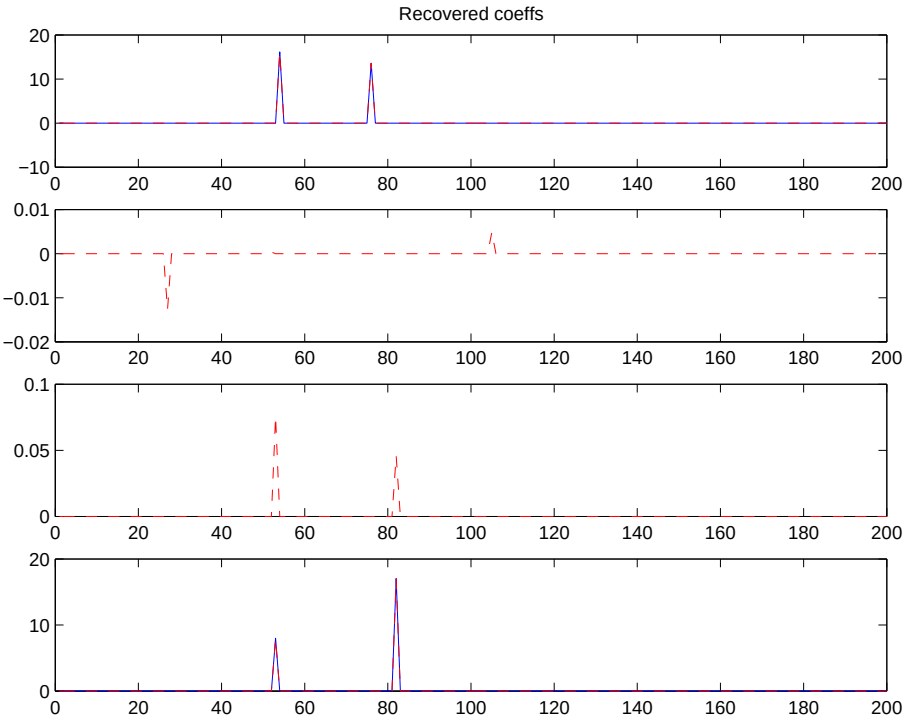

Figure 7.23: Original and recovered coefficients

Figure 7.23 shows the coefficients of the different signals.

3. **The analyzing dictionary $\mathcal{D}$ doesn't contain the waveforms in $\underline{s}$**

- *Synthesizing dictionary made of 5 α between 0 and 3 and analyzing dictionary made of 30 α between 0 and 3*

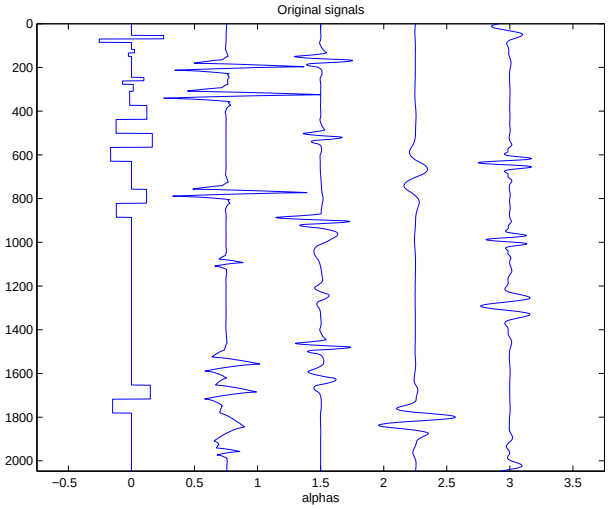

Figure 7.24: Original signals

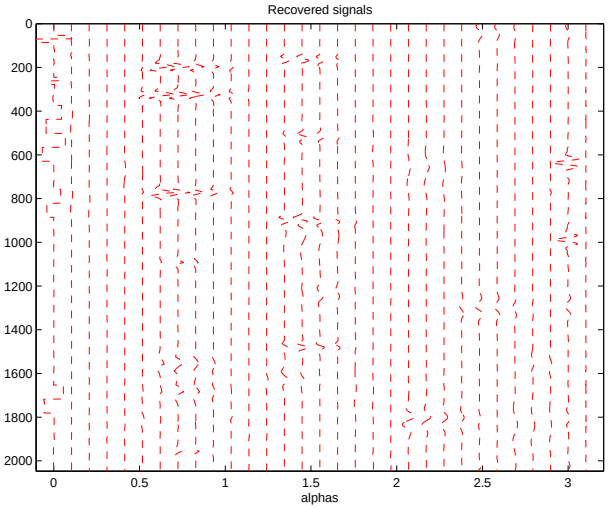

Figure 7.25: Recovered signals

Figures 7.24 and 7.25 illustrate the decomposition of a signal $\underline{s}$ containing 30 events, and show the original signals and the recovered ones. Figure 7.25 shows that $\underline{s}$ is separated in many more content types than it actually has, which implies that the events are not found appropriately. Not only can wrong events be selected, but new events can be created in the reconstructed signals.

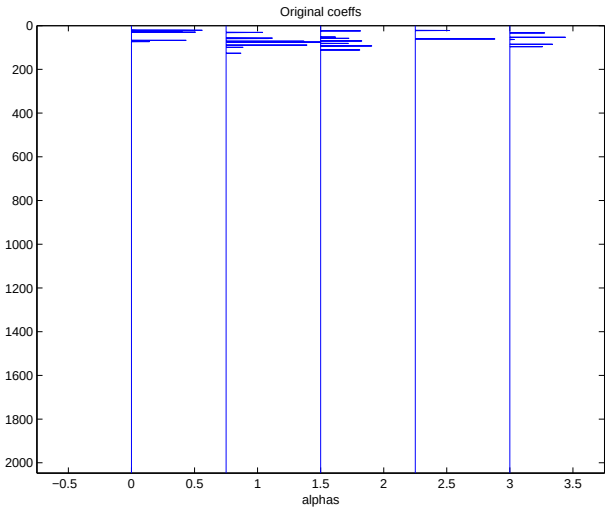

Figure 7.26: Original coefficients

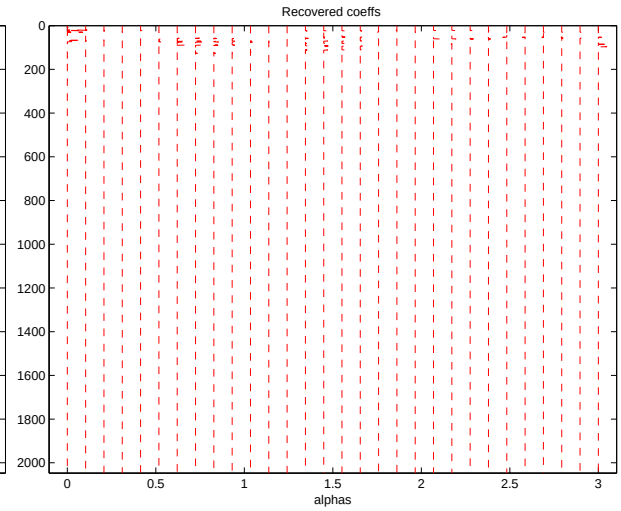

Figure 7.27: Recovered coefficients

Figures 7.26 and 7.27 show the same results for the coefficients.

- **The analyzing dictionary $\mathcal{D}$ doesn't contain the waveforms in $\underline{s}$**

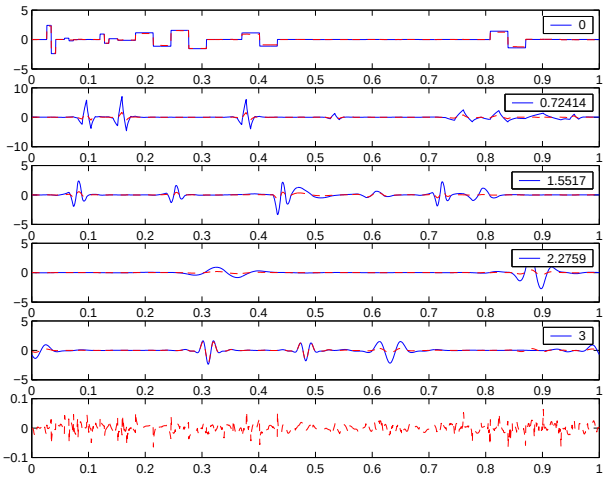

Figure 7.28: BCR separation with decimated fractional spline wavelets

Figure 7.28 shows the decomposition of the signal $\underline{s}$ with respect to a large dictionary whose fractional-orders are different from the fractional-orders contained in $\underline{s}$. Figure 7.28 shows the superposition of the original content types of $\underline{s}$ and the reconstructed ones with closest fractional-order. The four top plots show the original signals (in blue) and the recovered signals (in red) and the bottom plot shows the residual. Except for the zero-order signal (first plot) which is well recovered and the third order signal (fourth plot) which is partially recovered, the other content types of $\underline{s}$ aren't. The residual seems to contain most of the events. The failure of the decomposition is most probably due to the size of the analyzing dictionary I used, as well as the fact that the waveforms in $\underline{s}$ were not in the analyzing dictionary. The analyzing dictionary was fairly large and contained very similar waveforms, which reduced the efficiency of the BCR algorithm.

7.6.7 Discussion

The above results show the efficiency of the BCR algorithm in separating a signal $\underline{s}$ into different content types described by their singularity order α_i As explained earlier in this paper, the BCR algorithm can be very efficient if each dictionary used for each content type are very good in representing this content type and bad in representing the other ones. However, the more similar the atoms in the dictionaries, the less efficient the BCR algorithm. The decomposition we envision is already badly conditioned because the dictionaries we need to use contain similar atoms. However, in certain cases, the BCR algorithm succeeds in separating $\underline{s}$ into the correct content types.

First, there is a great importance in the decrease of the parameters δ_i in the algorithm. The slower the decrease, the better the separation, as the algorithm has time to make the difference between the various content types. The BCR algorithm is basically looking at the successive inner products between the signal $\underline{s}$ and the waveforms φ_i in $\mathbf{\Phi}_i$, and is putting the largest inner products (larger than the threshold δ_i) into the content type of order α_i. If δ_i decreases too fast, a large number of inner products corresponding to various orders will be larger than δ_i, and thus will be put wrongly into the content type of order α_i. There is also a concern about the dictionary. If the dictionary contains waveforms that are too similar or if the dictionary is too large (which also triggers very similar waveforms), it will be more difficult to distinguish between the different inner products, because they will have similar values. One solution could be to make the parameters δ_i decrease very slowly but that would make the algorithm prohibitively slow, and moreover would only work optimally if the analyzing and synthesizing dictionaries are the same. There is also a trade-off between the size of the dictionary and the fractional-orders α that we choose in $\mathcal{D}$. The number of nonzero entries in the original coefficient $\underline{c}$ is not important here because it is the dissimilarity between the different content types of $\underline{s}$, and the adequate choice of the dictionaries $\mathbf{\Phi}_i$ that make the BCR algorithm succeed.

From these studies on the performance of the BCR algorithm, we can conclude that if the analyzing dictionary $\mathcal{D}$ consists of waveforms that are in $\underline{s}$, the BCR algorithm will succeed or

partially succeed in separating $\underline{s}$ into the correct content types, provided $\mathcal{D}$ is not too large, and the fractional-orders in $\mathcal{D}$ not too close. However, when the dictionary contains atoms that are not in $\underline{s}$, the BCR algorithm starts having trouble in finding the correct decomposition. If we know a priori the waveforms in $\underline{s}$, it is of course better. Unfortunately this doesn't happen in practice because the waveforms in $\underline{s}$ are unknown. Therefore, we will never be able to construct a dictionary $\mathcal{D}$ that contains the waveforms in $\underline{s}$. In practice, we use fractional B splines, decimated or undecimated fractional spline wavelets.

The last issue of the BCR is its computational cost. In the BCR algorithm, the dictionaries need to be normalized. However, depending on the dictionary and its implementation, the normalization can become computationally intensive and the BCR algorithm prohibitively slow. This is for example the case for the undecimated fractional spline wavelet dictionary.

7.7 Conclusion

The studies of MP, BP and the BCR algorithm show that it is very difficult to estimate the fractional-orders of seismic signals by computing their representation in an overcomplete dictionary parameterized by their fractional-orders α. These algorithms work for very particular coefficients and well conditioned dictionaries. As soon as the dictionary is too large, and/or the waveforms too similar, and/or the coefficient vector not sparse enough, none of the above algorithms are able to find the right representation $\underline{c}$.

Real data falls under these considerations, as we need a large enough dictionary (thus with similar waveforms) to find the orders contained in $\underline{s}$. The estimated fractional-orders are limited by the fractional-orders contained in the dictionary, and the maximal error between the fractional-orders α found and the real ones in $\underline{s}$ is given by the largest distance between two consecutive fractional-orders contained in $\mathcal{D}$. To avoid large errors, it is therefore necessary to decompose $\underline{s}$ into a large dictionary, triggering the failure of MP, BP and the BCR algorithm. The representation $\underline{c}$ of seismic signals is not sparse enough for the type of dictionary we need.

Bibliography

[1] T. Blu and M. Unser. A complete family of scaling functions: the (α, τ) fractional splines. In *Proceedings of the Twenty-Eighth IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP'03)*, volume 6, pages 505–508, Hong Kong SAR, People's Republic of China, April 6-10 2003.

[2] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1999.

[3] I. Daubechies. Time-frequency localization operators: a geometric phase approach. *IEEE Transactions on Information Theory*, 34:605–612, 1988.

[4] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Comm. Pure Appli. Math*, 57:1413–1541, 2004.

[5] M.V. de Hoop and N. Bleistein. Generalized Radon transform inversions for reflectivity in anisotropic elastic media. *Inverse Probl.*, 13(3):669–690, 1997.

[6] F.J. Dessing. *A Wavelet Transform Approach to Seismic Processing*. PhD thesis, Delft University of Technology, 1997.

[7] D.L.Donoho and M. Elad. Optimally sparse representation in general (non orthogonal) dictionaries via l^1 minimization. *Proc. Nat. Acad. Sci.*, 100:2197–2202, 2003.

[8] P.E. Gill, W. Murray, D.B. Ponceleon, and M.A. Saunders. Solving the kkt systems in barrier methods for linear and quadratic programming. *Technical Report SOL 91-7*, July 1991.

[9] J. C. Harms and P. Tackenberg. Seismic signatures of sedimentation models. *Geophysics*, 37(1):45–58, February 1972.

[10] Felix J. Herrmann. Multiscale analysis of well- and seismic data. *Mathematical Methods in Geophysical Imaging V, SPIE*, 3453:180–208, 1998.

[11] Felix J. Herrmann. Fractional Spline Matching Pursuit: a quantitative tool for seismic stratigraphy. In *Soc. Expl. Geoph.*, 2001.

[12] Felix J. Herrmann. Multi- and monoscale attributes for well- and seismic data. *Chapter in the annual report of the Borehole Acoustics and Logging and Reservoir Delineation Consortia at MIT*, 2001.

[13] Felix J. Herrmann. Singularity Characterization by Monoscale Analysis: Applications to Seismic Imaging. *Applied and Computational Harmonic Analysis*, 11:64–88, 2001.

[14] Felix J. Herrmann. Seismic deconvolution by atomic decomposition: a parametric approach with sparseness constraints. *Integr. Computer-Aided Eng.*, 2005.

[15] Felix J. Herrmann and Yves Bernabé. Seismic singularities at upper mantle discontinuities: a site percolation model. *Geop. J. Int.*, 2004.

[16] Felix J. Herrmann, William J. Lyons, and Colin Stark. Seismic facies characterization by monoscale analysis. *Geophysical Research Letters*, 28(19):3781–3784, 2001.

[17] Felix J. Herrmann and Colin Stark. Monoscale analysis of edges/reflectors using fractional differentiations/integrations. In *Soc. Expl. Geoph.*, 1999.

[18] F.J. Herrmann. *A Scaling Medium Representation, a Discussion on Well-Logs, Fractals and Waves*. PhD thesis, Delft University of Technology, 1997.

[19] J. Liouville. Sur une formule pour les différentielles à indices quelconques à l'occasion d'un mémoire de M. Tortolini. *J. Math Pures Appl.*, unnumbered (in French), 1855.

[20] S. Mallat. A wavelet tour of signal processing. 1997.

[21] S. Mallat and Z. Zhang. Matching pursuit in a time-frequency dictionary. *IEEE Transactions on Signal Processing*, pages 3397–3415, 1993.

[22] J. Muller, I. Bokn, and J.L.McCauley. Multifractal analysis of petrophysical data. *Ann. Geophysicae*, 10:735–761, 1992.

[23] C. Payton. Seismic stratigraphy - applications to hyfrocarbon exploration. *AAPG*, 1977.

[24] S. Sardy, A.G. Bruce, and P.Tseng. Block Coordinate Relaxation Methods for Nonparametric Wavelet Denoising. *Journal of Computational and Graphical Statistics*, 9, 2000.

[25] J.L. Starck, M.Elad, and D.L.Donoho. Image Decomposition Via the Combination of Sparse Representation and a Variational Approach. *IEEE Transaction on Image Processing*.

[26] J.L. Starck, M.Elad, and D.L.Donoho. Redundant Multiscale Transforms and Their Application for Morphological Component Analysis. *Advances in Imaging and Electron Physics*, 132, 2004.

[27] M.T. Taner, F. Koehler, and R.E. Sheriff. Complex seismic trace analysis. *Geophysics*, 44(6):1041–1063, June 1979.

[28] M. Unser and T. Blu. Fractional Splines and Wavelets. *SIAM Review*, 42:43–67, 2000.

[29] C.P.A. Wapenaar. Amplitude-variations-with-angle behavior of self-similar interfaces. *Geophysics*, 64:1928, 1999.

[30] Meyer Yves. *Ondelettes et algorithmes concurrents*, volume 1435 of *Actualités Scientifiques et Industrielles [Current Scientific and Industrial Topics]*. Hermann, Paris, 1992.

Appendix A
Wavelets

Definition A.0.1 (Wavelet ,[20]) *A wavelet is a function $\psi \in L^2(\mathbb{R})$ of zero average:*

$$\int_{\mathbb{R}} \psi(t)dt = 0,$$

which is dilated with a scale parameter σ, and translated by a:

$$\psi_{(a,\sigma)}(t) = \frac{1}{\sqrt{\sigma}}\psi\left(\frac{t-a}{\sigma}\right).$$

Definition A.0.2 (Continuous Wavelet Transform, [20]) *The wavelet transform of a function $f \in L^2(\mathbb{R})$ at the scale σ and position a, $\mathcal{W}f(a,\sigma)$, is computed by correlating f with the wavelet at position a and scale σ:*

$$\begin{aligned} \mathcal{W}f(a,\sigma) &= \int_{-\infty}^{+\infty} f(t)\psi^*_{(a,\sigma)}(t)dt \\ &= \int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{\sigma}}\psi^*\left(\frac{t-a}{\sigma}\right)dt. \end{aligned}$$

Any signal $f \in L^2(\mathbb{R})$ can be written as a continuous summation of wavelets:

$$\forall\ t \in \mathbb{R}\ ,\ f(t) = K\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mathcal{W}f(a,\sigma)\psi_{(a,\sigma)}(t)\frac{d\sigma da}{\sigma},$$

where K is a constant.

This representation is very redundant and the coefficients $\mathcal{W}f(a,\sigma)$ are highly correlated. In order to reduce this redundancy, we can limit ourselves to dyadic wavelets and hence obtain a basis of $L^2(\mathbb{R})$.

Definition A.0.3 (Orthonormal Wavelet Bases, [21]) *Dyadic wavelets $\left\{\psi_{(j,k)} = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^jk}{2^j}\right)\right\}_{(j,k)\in\mathbb{Z}^2}$ generate an orthonormal basis of $L^2(\mathbb{R})$.*

Because the basis is orthonormal, the wavelet coefficients of a function $f \in L^2(\mathbb{R})$ *,* $\mathcal{W}f(j,k)$ *, are simply computed by taking the inner product between* f *and each wavelet:*

$$\begin{aligned} \mathcal{W}f(j,k) &= \langle f, \psi_{(j,k)} \rangle \\ &= \int_{-\infty}^{+\infty} f(t) \psi^*_{(j,k)} dt \\ &= \int_{-\infty}^{+\infty} f(t) \frac{1}{\sqrt{2^j}} \psi^* \left(\frac{t - 2^j k}{2^j} \right) dt. \end{aligned}$$

Any signal f *can be represented uniquely as:*

$$\forall\ t \in \mathbb{R}\ ,\ f(t) = \sum_{j=-\infty}^{+\infty} \sum_{k=-\infty}^{+\infty} \langle f, \psi_{(j,k)} \rangle \psi_{(j,k)}.$$

Appendix B
Bases

B.1 Fractional Splines

Proposition B.1.1 *Let $\alpha = 0$, and let $\varphi_0 = \varphi_{\alpha=0}$ be a vector of $\mathbb{R}^N$, such that:*

$$\forall\ 1 \leq n \leq N\ ,\ \varphi_0(n) = 1\ ,\ \textit{and}\ \forall\ n \leq 0\ ,\ \varphi_0(n) = 0.$$

Define:

$$\mathcal{D}_0 = \left\{v_k^0\right\}_{1\leq k\leq N} = \left\{v_k^0 \in \mathbb{R}^N\ ,\ 1 \leq k \leq N\ ,\ s.t.\ \forall\ 1 \leq n \leq N\ ,\ v_k^0(n) = \varphi_0(n-k+1)\right\}$$

$\mathcal{D}_0$ is a basis of $\mathbb{R}^N$

PROOF:

$\mathcal{D}_0$ consists of N vectors of $\mathbb{R}^N$, so we only need to prove that the vectors of $\mathcal{D}_0$ are either linearly independent or that $\mathbb{R}^N$ can represent any vector of $\mathbb{R}^N$. In this proof, we will prove that the vectors v_k^0 are linearly independent.

Suppose there exists N non zero numbers $\beta_1, ..., \beta_N$, such that:

$$\sum_{i=1}^{N} \beta_i v_i^0 = 0. \tag{B.1}$$

Equation (B.1) is equivalent to:

$$\forall\ 1 \leq n \leq N\ ,\ \sum_{i=1}^{N} \beta_i v_i^0(n) \ = \ 0 \tag{B.2}$$

$$\Leftrightarrow \forall\ 1 \leq n \leq N\ ,\ \sum_{i=1}^{N} \beta_i \varphi_0(n-i+1) \ = \ 0. \tag{B.3}$$

We now will prove recursively that all the β_i are zero .

Define the following property for an integer p, $1 \leq p \leq N$:

$$\forall\ 1 \leq j \leq p\ ,\ \beta_j = 0. \tag{B.4}$$

1. Let's show that this property is true for $p = 1$.

   If we evaluate equation (B.3) at $n = 1$, we obtain:

   $$\sum_{i=1}^{N} \beta_i \varphi_0(2-i) \quad = \quad 0, \tag{B.5}$$

   which is equivalent to:

   $$\beta_1 \varphi_0(1) = 0, \tag{B.6}$$

   as $\varphi_0(2-i)$ is only non zero for $i = 1$.

   As $\varphi_0(1) = 1$, we deduce that $\beta_1 = 0$, which is property (B.4) for $p = 1$.

2. Now, we will show that if property (B.4) is satisfied for an integer p between 1 and N, it is also satisfied for $p+1$. Let p be an integer, $1 \leq p \leq N$ and suppose that property (B.4) is true for p, i.e.:

   $$\forall\ 1 \leq j \leq p\ ,\ \beta_j = 0.$$

   We have:

   $$\sum_{i=p+1}^{N} \beta_i v_i^0 = 0 \tag{B.7}$$

   $$\Leftrightarrow \forall\ 1 \leq n \leq N\ ,\ \sum_{i=p+1}^{N} \beta_i \varphi_0(n-i+1) \quad = \quad 0 \tag{B.8}$$

   If we evaluate equation (B.22) at $n = p+1$ we obtain:

   $$\sum_{i=p+1}^{N} \beta_i \varphi_0(p-i+2) = 0, \tag{B.9}$$

which is equivalent to:

$$\beta_{p+1}\varphi_0(p+1) = 0, \tag{B.10}$$

as $\varphi_0(p-i+2)$ is only non zero for $i = p+1$.

As $\varphi_0(p+1) = 1$, we deduce that $\beta_{p+1} = 0$, which is property (B.4) for $p+1$.

3. Property (B.4) is true for $p = 1$. If it is true for p, it is also true for $p+1$, so it is true for all $1 \leq n \leq N$, i.e. $\forall\ 1 \leq n \leq N$, $\beta_n = 0$.

Conclusion:

v_k^0 for $1 \leq k \leq N$, are linearly independent and $\mathcal{D}_0$ is a basis of $\mathbb{R}^N$

Proposition B.1.2 *Let $\alpha > 0$, and let φ_α be a vector of $\mathbb{R}^N$, such that:*

$$\forall\ 1 \leq n \leq N\ ,\ \varphi_\alpha(n) = n^\alpha + 1\ ,\ \textit{and}\ \forall\ n \leq 0\ ,\ \varphi_\alpha(n) = 1.$$

Define:

$$\mathcal{D}_\alpha = \left\{v_k^\alpha\right\}_{1\leq k\leq N} = \left\{v_k^\alpha \in \mathbb{R}^N\ ,\ 1 \leq k \leq N\ ,\ s.t.\ \forall\ 1 \leq n \leq N\ ,\ v_k^\alpha(n) = \varphi_\alpha(n-k+1)\right\}$$

$\mathcal{D}_\alpha$ is a basis of $\mathbb{R}^N$

PROOF:

This proof is very similar to the previous one. $\mathcal{D}_\alpha$ has N vectors in $\mathbb{R}^N$ so it is sufficient to prove that the vectors v_k^α are linearly independent.

Suppose there exists N non zero numbers $\beta_1, ..., \beta_N$, such that:

$$\sum_{i=1}^{N} \beta_i v_i^\alpha = 0 \tag{B.11}$$

$$\Leftrightarrow \forall\ 1 \leq n \leq N\ ,\ \sum_{i=1}^{N} \beta_i v_i^\alpha(n) = 0 \tag{B.12}$$

$$\Leftrightarrow \forall\ 1 \leq n \leq N\ ,\ \sum_{i=1}^{N} \beta_i \varphi_\alpha(n-i+1) = 0. \tag{B.13}$$

If we evaluate equation (B.13) at $n = 1, n = 2, ..., n = N$, we obtain the following N equations:

$$\sum_{i=1}^{N} \beta_i = 0 \tag{B.14a}$$

$$\varphi_\alpha(2)\beta_1 + \sum_{i=2}^{N} \beta_i = 0 \tag{B.14b}$$

$$\varphi_\alpha(2)\beta_1 + \varphi_\alpha(3)\beta_2 \sum_{i=3}^{N} \beta_i = 0 \tag{B.14c}$$

$$...$$

$$\varphi_\alpha(2)\beta_1 + \varphi_\alpha(3)\beta_2 + ... + \varphi_\alpha(N)\beta_{N-1} + \beta_N = 0. \tag{B.14d}$$

We will also use a recursive argument, using property (B.4).

1. Let's show that property (B.4) is true for $p = 1$.

   Combining equations (B.14a) and (B.14b), we obtain: $(\varphi_\alpha(2) - 1)\beta_1 = 0$, which implies $\beta_1 = 0$ because $\varphi_\alpha(2) \neq 1$. So property (B.4) is true for $p = 1$.

2. Now, let p be an integer, $1 \leq p \leq N$ and suppose that property (B.4) is true for p, i.e.:

   $$\forall\ 1 \leq j \leq p\ ,\ \beta_j = 0.$$

   With this property, the set of N equations becomes:

   $$\sum_{i=p+1}^{N} \beta_i = 0 \qquad \text{(B.15a)}$$

   $$\varphi_\alpha(p+2)\beta_{p+1} + \sum_{i=p+2}^{N} \beta_i = 0 \qquad \text{(B.15b)}$$

   $$\varphi_\alpha(p+2)\beta_{p+1} + \varphi_\alpha(p+3)\beta_{p+2} \sum_{i=p+3}^{N} \beta_i = 0 \qquad \text{(B.15c)}$$

   $$\ldots$$

   $$\varphi_\alpha(p+2)\beta_{p+1} + \varphi_\alpha(p+3)\beta_{p+2} + \ldots + \varphi_\alpha(N)\beta_{N-1} + \beta_N = 0. \qquad \text{(B.15d)}$$

   As for $p = 1$, combining equations (B.15a) and (B.15b) gives us $(\varphi_\alpha(p+2) - 1)\beta_{p+1} = 0$ which implies that $\beta_{p+1} = 0$ because $\varphi_\alpha(p+2) \neq 1$. So property (B.4) is true for $p + 1$.

3. $\forall\ 1 \leq n \leq N\ ,\ \beta_n = 0.$

Conclusion:

v_k^α for $1 \leq k \leq N$, are linearly independent and $\mathcal{D}_\alpha$ is a basis of $\mathbb{R}^N$

B.2 Fractional Derivatives of Gaussians

Let α be a positive fractional number, and let g_α be the α-th derivative of a Gaussian. A Gaussian doesn't have a compact support, but if it decays fast enough, we can approximate it by a function that has a compact support. If we sample g_α, we obtain a vector of length N

that we call u_α. Now let's define $\mathcal{S}$ as the set of integers m where the vector u_α is less than a tolerance (let's say $tol = 10^{-3}$, but this can be changed depending on the applications):

$$\mathcal{S} = \Big\{ m \in \mathbb{N} \text{ , such that } for\ 1 \leq m \leq N \text{ , } |u_\alpha(m)| < tol \Big\}$$

and let's put these coefficients to zero:

$$\forall\ n \in \mathcal{S}\ ,\ u_\alpha(n) = 0$$

Define the vector φ_α, as the first circular shifted vector v, such that $v(1) \neq 0$ but $v(0) = v(N) = 0$, i.e.:

$$\varphi_\alpha = \Big\{ v \in \mathbb{R}^N \text{ , such that } \exists\ l \in \mathbb{N}\ ,\ \forall\ 1 \leq n \leq N\ ,\ v(n) = \mathcal{S}^l u_\alpha\ ,\ v(1) \neq 0 \text{ and } v(N) = 0 \Big\},$$

where $\mathcal{S}$ is the circular shift operator.

By construction, $\forall\ n > N$ and $n \leq 0$, $\varphi_\alpha(n) = 0$.

Proposition B.2.1 *Define:*

$$\mathcal{D}_\alpha = \Big\{ v_k^\alpha \Big\}_{1 \leq k \leq N} = \Big\{ v_k^\alpha \in \mathbb{R}^N\ ,\ 1 \leq k \leq N\ ,\ s.t.\ \forall\ 1 \leq n \leq N\ ,\ v_k^\alpha(n) = \varphi_\alpha(n-k+1) \Big\}.$$

$\mathcal{D}_\alpha$ *is a basis of* $\mathbb{R}^N$

PROOF:

The proof is similar to the proof of the fractional splines for $\alpha = 0$, because, for $2 \leq k \leq N$, each vector v_k^α has one more zero component than the previous vector v_{k-1}^α. Similarly to the fractional splines, we only need to prove that the vectors v_k^α are linearly independent and we will prove it recursively.

If we suppose there exists N non zero numbers $\beta_1, ..., \beta_N$, such that:

$$\sum_{i=1}^{N} \beta_i v_i^\alpha = 0 \tag{B.16}$$

$$\Leftrightarrow \forall\ 1 \leq n \leq N\ ,\ \sum_{i=1}^{N} \beta_i v_i^\alpha(n) = 0 \tag{B.17}$$

$$\Leftrightarrow \forall\ 1 \leq n \leq N\ ,\ \sum_{i=1}^{N} \beta_i \varphi_\alpha(n-i+1) = 0. \tag{B.18}$$

1. Property for $p = 1$

   If we evaluate equation (B.18) at $n = 1$, we get:

$$\sum_{i=1}^{N} \beta_i \varphi_0(2-i) \quad = \quad 0, \tag{B.19}$$

   which is equivalent to:

$$\beta_1 \varphi_0(1) = 0, \tag{B.20}$$

   as $\varphi_0(2-i)$ is only non zero for $i = 1$.

   As $\varphi_0(1) \neq 0$, we deduce that $\beta_1 = 0$, which is property (B.4) for $p = 1$.

2. Now, we will show that if property (B.4) is satisfied for an integer p between 1 and N, it is also satisfied for $p + 1$.

   Let p be an integer, $1 \leq p \leq N$ and suppose that property (B.4) is true for p, i.e.:

$$\forall\ 1 \leq j \leq p\ ,\ \beta_j = 0$$

   So we have:

$$\sum_{i=p+1}^{N} \beta_i v_i^0 = 0 \tag{B.21}$$

$$\Leftrightarrow \forall\ 1 \leq n \leq N\ ,\ \sum_{i=p+1}^{N} \beta_i \varphi_0(n-i+1) \quad = \quad 0. \tag{B.22}$$

   If we evaluate equation (B.22) at $n = p + 1$ we obtain:

$$\sum_{i=p+1}^{N} \beta_i \varphi_0(p-i+2) = 0, \tag{B.23}$$

   which is equivalent to:

$$\beta_{p+1} \varphi_0(p+1) = 0, \tag{B.24}$$

   as $\varphi_0(p-i+2)$ is only non zero for $i = p+1$. As $\varphi_0(p+1) \neq 0$, we deduce that $\beta_{p+1} = 0$, which is property (B.4) for $p + 1$.

3. Property (B.4) is true for $p = 1$. If it is true for p, it is also true for $p+1$, so it is true for all $1 \leq n \leq N$, i.e. $\forall\ 1 \leq n \leq N$, $\beta_n = 0$.

Conclusion:

v_k^α for $1 \leq k \leq N$, are linearly independent and $\mathcal{D}_\alpha$ is a basis of $\mathbb{R}^N$

Appendix C
Soft Thresholding

Proposition C.0.2 (Soft Thresholding) *Let $a, y \in \mathbb{R}$ and $\lambda \in \mathbb{R}_+^*$, then:*

$$sign(a)(|a| - \frac{1}{2\lambda})_+ = arg\min_x(\lambda(a-x)^2 + |x|),$$

where $y_+ = \max(0, y)$, for $y \in \mathbb{R}$.

PROOF:

Let $f(x) = |x| + \lambda(a-x)^2$, and let's call $\hat{x} = arg\min_x f(x)$.

We have:

$$f(x) = \begin{cases} x + \lambda(a-x)^2 & \text{if } x > 0 \\ \lambda a^2 & \text{if } x = 0 \\ -x + \lambda(a-x)^2 & \text{if } x < 0 \end{cases}$$

and

$$f'(x) = \begin{cases} 1 - 2\lambda(a-x) & \text{if } x > 0 \\ -1 - 2\lambda(a-x) & \text{if } x < 0 \end{cases}$$

f is not differentiable at 0 because $f'(0_+) = 1 - 2\lambda a$ and $f'(0_-) = -11 - 2\lambda a$.

Let $x \neq 0$:

$$f'(x) = 0 \iff \begin{cases} x = a - \frac{1}{2\lambda} \text{ if } x > 0 \quad \text{i.e.} \quad a > \frac{1}{2\lambda} \\ x = a + \frac{1}{2\lambda} \text{ if } x < 0 \quad \text{i.e.} \quad a < -\frac{1}{2\lambda} \end{cases} \tag{C.1}$$

Equation (C.1) can be rewritten as:

$$x = sign(a)(|a| - \frac{1}{2\lambda}) \text{ if } |a| > \frac{1}{2\lambda} \tag{C.2}$$

Now, if $|a| \leq \frac{1}{2\lambda}$, then f' can never be zero, and hence the minimum of f cannot happen when the derivative is zero.

Now, consider $|a| \leq \frac{1}{2\lambda}$ and note that:

$$\begin{aligned} |a| \leq \frac{1}{2\lambda} &\iff -\frac{1}{2\lambda} \leq a \leq \frac{1}{2\lambda} \\ &\iff -1 \leq -2\lambda a \leq 1 \\ &\iff 1 - 2\lambda a \geq 0 \text{ and } -1 - 2\lambda a \leq 0. \end{aligned}$$

Recall the expression of f':

$$f'(x) = \begin{cases} 1 - 2\lambda a + 2\lambda x & \text{if } x > 0 \\ -1 - 2\lambda a + 2\lambda x & \text{if } x < 0 \end{cases}$$

- On $\mathbb{R}_+^*$, $f'(x) = 1 - 2\lambda a + 2\lambda x$ where $1 - 2\lambda a \geq 0$ and $2\lambda x > 0$, so the sign of f' is given by the y-intercept $(1 - 2\lambda)$. So on $\mathbb{R}_+^*$, $f'(x) \geq 0$.
- In the same way on $\mathbb{R}_-^*$, $f'(x) = -1 - 2\lambda a + 2\lambda x$ where $-1 - 2\lambda a \leq 0$ and $2\lambda x > 0$, so the sign of f' is given by the y-intercept $(-1 - 2\lambda)$. So on $\mathbb{R}_-^*$, $f'(x) \leq 0$.

So f, defined on $\mathbb{R}$, is decreasing on $\mathbb{R}_-^*$, and increasing on $\mathbb{R}_+^*$ which means that the minimum of f is at zero. Summarizing the results, we get:

$$\begin{cases} \text{If } |a| > \frac{1}{2\lambda} \text{ then } \hat{x} = sign(a)(|a| - \frac{1}{2\lambda}) \\ \text{If } |a| \leq \frac{1}{2\lambda} \text{ then } \hat{x} = 0, \end{cases} \tag{C.3}$$

which can be written as one equation:

$$\boxed{\hat{x} = sign(a)(|a| - \tfrac{1}{2\lambda})_+}$$

Appendix D
Fourier Transform and Fast Fourier Transform

This appendix explains how to compute the Fourier Transform of a function via the fft.

D.1 Definitions

Definition D.1.1 (Fourier Transform) *Let $f \in \mathbb{L}^1(\mathbb{R})$. The Fourier Transform of f is defined by:*

$$\forall n \in \mathbb{Z}, \quad c_n(f) = \frac{1}{T}\int_0^T f(t)\mathrm{e}^{-2i\pi n\frac{t}{T}}\,dt$$

The Fourier Transform (FT) of f is the map defined on $\mathbb{R}$:

$$x \mapsto \mathrm{FT}f(x) = \int_{\mathbb{R}} f(t)\mathrm{e}^{-2i\pi xt}\,dt$$

Definition D.1.2 (Fourier Series) *Let f be a T-periodic function. The Fourier coefficients are defined as:*

$$\forall n \in \mathbb{Z}, \quad c_n(f) = \frac{1}{T}\int_0^T f(t)\mathrm{e}^{-2i\pi n\frac{t}{T}}\,dt$$

It is the sequence $(c_n(f))_{n\in\mathbb{Z}}$, indexed by $\mathbb{Z}$.

Definition D.1.3 (Fast Fourier Transform (FFT**))** *Let f be a T-periodic function. The Fourier coefficients computed by the* FFTare given by:

$$\forall 1 \leq n \leq N \text{ , } \mathrm{FFT(f)} = \frac{1}{\mathrm{N}}\sum_{\mathrm{k}=0}^{\mathrm{N}-1}\mathrm{f}(\frac{\mathrm{kT}}{\mathrm{N}})\omega_{\mathrm{N}}^{-\mathrm{nk}}$$

D.2 Approximation of the Fourier coefficients with the FFT

Let f be a T-periodic function. Its Fourier coefficients are given by :

$$\forall n \in \mathbb{Z}, \quad c_n(f) = \frac{1}{T}\int_0^T f(t)\mathrm{e}^{-2i\pi n\frac{t}{T}}\,dt$$

We can approximate the above integral by using the lower sum:

$$\frac{T}{N}\sum_{k=0}^{N-1} f(\frac{kT}{N})\mathrm{e}^{-2i\pi n\frac{kT}{NT}} = \frac{T}{N}\sum_{k=0}^{N-1} f(\frac{kT}{N})\omega_N^{-nk},$$

where $\omega_N = \mathrm{e}^{\frac{2i\pi}{N}}$ racine N-ième de l'unité.

An approximate value for the Fourier coefficients $c_n(f)$ is therefore:

$$\frac{1}{N}\sum_{k=0}^{N-1} f(\frac{kT}{N})\omega_N^{-nk} = \mathrm{FFT(f)}$$

D.3 Fourier coefficients and Fourier Transform

Let $f \in \mathbb{L}^1(\mathbb{R})$, we set

$$\varphi(t) = T\sum_{n\in\mathbb{Z}} f(t+nT),$$

where $\varphi \in \mathbb{L}^1(\mathbb{R}/T\mathbb{Z})$.

Computing the Fourier coefficients of φ gives

$$c_n(\varphi) = \int_0^T \sum_{k\in\mathbb{Z}} f(t+kT)\mathrm{e}^{-2i\pi n\frac{t}{T}}\,dt = \int_{\mathbb{R}} f(t)\mathrm{e}^{-2i\pi n\frac{t}{T}}\,dt = \mathrm{FT}f(\frac{n}{T}).$$

The sampling is obtained by calculating the Fourier Transform of the function φ. We can therefore interpret the sequence $\left(\mathrm{FT}f(\frac{n}{T})\right)_{n\in\mathbb{Z}}$ as a sampling of the Fourier Transform of f. This sampling is periodic since the sequence $(\mathrm{FFTc}, \varphi, \mathrm{n},)_{\mathrm{n}}$ is N-périodic.

D.4 Fourier Transform and Fast Fourier Transform

To approximate the Fourier Transform of a function f with the FFT, f needs to be a continuous function and have a compact support [30]:

$$f \in C^0(\mathbb{R}) \text{ and } \forall |x| > T_0 \ , \ f(x) = 0.$$

Let's choose $T_1 > T_0$, and consider the function f as the restriction of a function $\tilde{f}$ to $[-T_1, T_1]$

$$f = \tilde{f} \restriction_{[-T_1,T_1]} .$$

where $\tilde{f}$ is $2T_1$ periodic.

Since $\tilde{f}$ is $2T_1$ periodic, we can compute its Fourier coefficients c_k:

$$c_k = \frac{1}{2T_1}\int_{-T_1}^{T_1} f(x)\exp(-i\pi kx/T_1) = \frac{1}{2T_1}\mathrm{FT}\left(\frac{k}{2T_1}\right) \tag{D.1}$$

When $T_1 \gg T_0$, the sampling defined above is sufficiently precise for our purposes. When f is not compactly supported, its Fourier Transform can be approximated by the Discrete Cosine Transform.

D.5 Approximation of the Fourier Transform in Matlab

In Matlab, the FFTof a vector $\underline{\mathrm{v}}$ (noted $\underline{\mathrm{u}}$) of length N, is defined as:

$$\mathrm{FFT}(\underline{\mathrm{v}}) = \underline{\mathrm{u}}(\mathrm{k}) = \sum_{\mathrm{n}=1}^{\mathrm{N}} \underline{\mathrm{v}}(\mathrm{n}))\mathrm{e}^{-\mathrm{i}2\pi(\mathrm{k}-1)(\mathrm{n}-1)/\mathrm{N}}, \text{ for } 1 \leq \mathrm{k} \leq \mathrm{N}$$

The inverse Fourier transform is given by:

$$\mathrm{IFFT}(\underline{\mathrm{u}}) = \underline{\mathrm{v}}(\mathrm{n}) = \sum_{\mathrm{k}=1}^{\mathrm{N}} \underline{\mathrm{u}}(\mathrm{k}))\mathrm{e}^{\mathrm{i}2\pi(\mathrm{k}-1)(\mathrm{n}-1)/\mathrm{N}}, \text{ for } 1 \leq \mathrm{n} \leq \mathrm{N}$$

Let f be a continuous function with compact support $[-T_0T_0]$, i.e. $f(x) = 0$ for $|x| > T_0$. Let $T_1 > T_0$. From (D.1) [30], we have:

$$c_n(f) = \frac{1}{2T_1}\mathrm{FT}\left(\frac{n}{2T_1}\right) \text{ for } n \in \mathbb{Z}$$

We can also approximate the Fourier coefficients of f by:

$$c_n(f) = \frac{1}{N}\sum_{k=1}^{N} x_k \mathrm{e}^{-2\pi(k-1)(n-1)/N}, \text{ for } n \in \mathbb{Z}$$

where $x_k = f(\frac{2(k-1)T_1}{N})$ for $1 \leq k \leq N$, and $2T_1$ is the period of the function. The Fourier coefficients are N-periodic, leading to $c_{n+pN} = c_N$ for $p \in \mathbb{Z}$.

If we take N large enough to have a good approximation of the Fourier coefficients, we can write:

$$c_n(f) \approx \frac{1}{N}\mathrm{FFT(f)}$$

where $f_k = f(\frac{2(k-1)T_1}{N})$ for $1 \leq k \leq N$.

Now, $c_n(f) \approx \frac{1}{N}\mathrm{FFT(f)}$ and $c_n(f) = \frac{1}{2T_1}\mathrm{FT}(\frac{n}{2T_1})$, which leads to $c_n(f) = \frac{1}{2T_1}\mathrm{FT}(\frac{n}{2T_1}) \approx \frac{1}{N}\mathrm{FFT(f)}$.

We can therefore write:

$$\frac{2T_1}{N}\mathrm{FFT(f)} \approx \mathrm{FT}(\frac{\mathrm{n}}{2\mathrm{T}_1})$$

for $1 \leq n \leq N$ or $-N/2+1 \leq n \leq N/2$.

The above formula gives an approximation of the Fourier Transform of a function f with the FFT.

Appendix E
Explanation of the software

README

`SeisCharac1_4.m` is the program that locates the reflectors and estimates their fractional-orders. The progam works for any size of the data (matrix or cube).

1. `BCRDaubechies1_2.m` Version 1.2

   Iterative thresholding algorithm with Coordinate Relaxation Method

```
function[signoise,coeffrec] = BCRDaubechies1_2(s,NameOfDict,par1,par2,par3,beta,czero,
Lmax,limit,expl,speed1,speed2)}

function[signoise,coeffrec] = BCRDaubechies1_2(s,NameOfDict,par1,par2,par3,beta,Lmax,
TOL,limit,expl,czero,speed1,speed2)

Solve the following minimization problem:
min beta1||c1||1 + ... + betan||cn||1 + (||s-s1-...-sn||2)^2
using the Block Coordinate Relaxation Method with respect
to the coefficients

Block Relaxation Coordinate method

INPUTS:
s:                  Signal we want to decompose
NameOfDict:         Name of the merged dictionary
par1,par2,par3:     Parameter of the merged dictionary
beta :              Parameter of the minimization (vector)
Lmax:               Max. # of iterations
czero:              Inital guess for the coefficients
limit:              Stopping criterion for the algorithm
expl:               0 -> linear decrease
                    1 -> exponential decrease
speed1:             Speed of linear/exponential decrease
speed2:             Speed of the power decrease (for exponential
```

```
                decrease only)

OUTPUTS:
signoise:       Matrix whose columns contain the different content
                parts as well as the residual in the last column
coeffrec:       Matrix whose column contain the coefficients of the
                different content parts
```

2. BuildDic1_3.m Version 1.3

Build the detection and estimation dictionaries

```
function[] = BuildDic1_3(whichdic,n,pardic_ave,savedetectest,nb_alpha,nb_sigma,
dist_alpha,percent_sigma)

Build the dictionaries for the estimation and detection and saves it
into:

INPUTS:
whichdic:          String: values -> 'detection' or 'both'
                   1- if whichdic = 'detection' -> only computes the
                   detection dictionary
                   1- if whichdic = 'both' -> computes the detection &
                   estimation dictionaries
n:                 Length of the signal
pardic_ave:        Average parameters of the different traces
                   1- pardic_ave(1) contains the average fractional
                   order of the traces
                   2- pardic_ave(2) contains the average std deviation
                   of the traces
savedetectest      Name of the detection or both dictionaries
nb_alpha:          Vector of length 2 containing the # of frac orders in the
                   detection and estimation dictionaries
                   1- nb_alpha(1): # of frac orders in the detection dic
                   2- nb_alpha(2): # of frac orders in the estimation dic
nb_sigma:          Vector of length 2 containing the # of std deviations in the
                   detection and estimation dictionaries
                   1- nb_sigma(1): # of std deviations in the detection dic
                   2- nb_sigma(2): # of std deviations in the estimation dic
dist_alpha:        Vector of length 2 containing the distances betweent
                   the fractional orders for the detection and
                   estimation dictionaries
                   1- dist_alpha(1): for the detection dic
                   2- dist_alpha(2): for the estimation dic
percent_sigma:     Maximum percentage of the average std deviation added to the
                   average std deviation. sigma_ave(1+/-
                   percent_sigma) will be the furthest std deviation
                   allowed in the dictionary
```

3. BuildFracGauss_dic1_2 Version 1.2

   Build a dictionary with fractional derivatives of the Gaussian

```
function[dict,par1,par2,par3,norm_dic,alpha_dic,sigma_dic,loopsigma] =
BuildFracGauss_dic1_2(n,alpha_ave,sigma_ave,dist_alpha,percentage,nb_alpha,nb_sigma)

function[dict,par1,par2,par3,norm_dic,alpha_dic,sigma_dic,loopsigma] =
BuildFracGauss_dic1_2(n,alpha_ave,sigma_ave,dist_alpha,percentage,nb_alpha,nb_sigma)

Build a merged dictionary with fractional derivatives of gaussians

 INPUTS:
n:                  Length of the signal
alpha_ave:          Average alpha in the signal
sigma_ave:          Average sigma in the signal
dist_alpha:         Distance between the alphas in the dictionary
percentage:         Maximum percentage of the average sigma
                    (~ frequency) added or subtracted from the average sigma
nb_alpha:           Nb of alphas in the dictionary
nb_sigma:           Nb of sigmas in the dictionary

OUTPUTS:
dict:               Name of the dictionary (merged dictionary)
par1,par2,par3:     Parameters of the dictionary
norm_dic:           Norm of the matrix of the dictionary
alpha_dic:          Vector of alphas in the dictionary
sigma_dic:          Vector of sigmas in the dictionary
loopsigma:          Flag:
                    = 1 if loop on sigma to build the dictionary is the 1st one
                    = 0 otherwise

See also MakeFracGauss1_2.m, MakeDic.m, NormFracGauss.m,
```

4. `chunk1_1.m` Version 1.1

   Process and analyze the output of the iterative thresholding algorithm

```
function[chunks_c,chunks_index,nbchunk,NbOfChunks,ind] = chunk1_1(c,tol)

Chunks the coefficient vector c into subgroups of events

INPUTS:
c:              Coefficient vector c
tol:            Threshold level (keep moduli of c that are larger than
                tol). tol is called mu in the paper.

OUTPUT:
chunks_c:       Cell array containing the different chunks of c (has
                NbOfChunks cells)
chunks_index:   Cell array containing the different chunks of indices
                of c (has NbOfChunks cells)
nbchunk:        Vector that keeps track of which chunk the indices are put in
                Usage: nbchunk(find(ind==index)) gives the nb of the chunk
                index is in.
NbOfChunks:     # of chunks found
ind             Indices of the moduli of c larger than tol
```

5. Estimate1_1.m Version 1.1

Estimate the fractional-orders of the detected events

```
function[v_alpha] = Estimate1_1(s,cmat,cfold,tolchunk,distblob,
alpha_dic,sigma_dic,loopalpha,table)

Estimate the fractional order alpha of the different events

INPUTS:
s:                      Signal analyzed
cmat:                   Matrix of the coefficient (output of BCRDaubechies)
cfold:                  Vector of the folded coefficient cmat (length(cfold)=n)
tolchunk:               Tolerance for the function chunk (keep the moduli
                         larger than tolchunk)
distblob:               Minimum distance between the chunks allowed (in GetBlob.m)
alpha_dic:              Vector of the fractional orders for the dictionary in
                        which cmat is decomposed in
sigma_dic:              Vector of the frequencies for the dictionary in which
                        cmat is decomposed in
loopalpha:              Flag:
                        = 1 if loop on alpha to build the dictionary is the 1st one
                        = 0 otherwise
table:                  Vector giving the locations in the signal of
                        the coefficient corresponding to a particular
                        atom in FracGauss dictionaries

OUTPUT:
v_alpha:                Vector whose nonzero entries give the
                        fractional order of the signal at this
                        particular location. Its length is the length
                        of the signal

See also chunk1_1.m, Getblob1_1.m, TableFracGauss1_2.m, whichalpha1_1.m,
BCRDaubechies1_2.m
```

6. `Fold1_1.m` Version 1.1

   Reshape the output of the iterative thresholding algorithm

```
function[cfold] = Fold1_1(c,n,NameOfDict,par1,par2,par3)

Fold the vector c unto the vector cfold'

INPUTS:
c:                Vector c of length NumberOfDicts*n
n:                Length of the signal
NameOfDict:       Name of the merged dictionary
par1,par2,par3:   Parameters of the dictionary

OUTPUT:
cfold:            Folded vector of length n by 1 obtained by summing the
                  absolute value of the different coefficients obtained
                  with the different wavelets in NameOfDict

See also SeisCharac1_4.m
```

7. `Getblob1_1.m` Version 1.1

   Process the output of `chunk1_1.m` for the analysis of the coefficient vector obtained from the iterative thresholding algorithm

```
function[blobs_c,blobs_index,nbblob,NbOfBlobs,indblob] =
Getblob1_1(c,chunks_c,chunks_index,nbchunk,NbOfChunks,dist)

Get the coefficients and put them into blobs

INPUTS:
c:              Coefficient vector
chunks_c:       Cell array containing the different chunks of c (has
                NbOfChunks cells)
chunks_index:   Cell array containing the different chunks of indices
                of c (has NbOfChunks cells)
nbchunk:        Vector that keeps track of which chunk the indices are put in
                Usage: nbchunk(find(ind==index)) gives the nb of the chunk
                index is in.
NbOfChunks:     # of chunks found
dist:           Minimum distance between the chunks allowed

OUTPUTS:
blobs_c:        Cell array containing the different blobs (merged
                chunks) of c (has NbOfBlobs cells)
blobs_index:    Cell array containing the different blobs of indices
                of c (has NbOfBlobs cells)
nbblob:         Vector that keeps track of which blob the indices are put in
                Usage: nbblob(find(ind==index)) gives the nb of the blob
                index is in.
NbOfBlobs:      # of blobs found
indblob:        Indices of the entries of c which are in the different blobs

See also chunk1_1.m
```

8. InitSeisCharac.m

   Give examples of parameters for the algorithm

```
function[par_detection,par_window,par_estimation,nb_alpha,nb_sigma,dist_alpha
,percent_sigma,par_bcr_d,par_bcr_e] = InitSeisCharac(n,ratio)

Initialize the default parameters for SeisCharac1_3.m

INPUTS:
n:                Length of the signal
ratio:            Ratio between the initial length of the traces and
                  the length of the traces currently analyzed

OUTPUTS:
par_detection:    Vector of length 2 containing the parameters for the
                  detection
                  1- par_detection(1): percentage of the norm
                  of the coefficient => par_detection(1)*maxval is the
                  threshold level for the detection
                  2- par_detection(2): minimum width of the sets of
                  coefficients associated to one event
par_window:       Vector of length 2 containing the parameters for the
                  windowing
                  1- par_window(1): # of sigma we want for half the
                  width of the window
                  2- par_window(2): # of sigma we want for the width of
                  the decay of the window
par_estimation:   Vector of length 2 containing the parameters for the
                  estimation (see par_detection)
par_bcr(_d;_e):   Vector of length 3 containing the parameters for the
                  BCR for detection and estimation (_d or _e)
                  1- par_bcr(1): beta
                  2- par_bcr(2): if =1 -> exponential decrease
                                    =0 -> linear decrease
                  3- par_bcr(3): speed1 for Block Solver (BCRDaubechies1_2.m)
                  4- par_bcr(4): speed2 for Block Solver (BCRDaubechies1_2.m)

See also SeisCharac1_4.m
```

9. NormFracGauss.m

   Compute the norm of the dictionary made of fractional derivatives of the Gaussian.

```
function[norme] = NormFracGauss(n,dict,par1,par2,par3)

Computes the norm of the FracGauss dictionary

INPUTS:
n:                Length of the signal
dict:             Name Of the dictionary
par1,par2,par3:   Parameters of the dictionary

OUTPUT:
norme:            Norm of the matrix of the dictionaray

See also SeisCharac1_4.m, SizeOfDict.m, FastS.m,
FastA.m
```

10. `PrecomputeDict1_2.m` Version 1.2

Precompute a table of estimation dictionaries

```
function[dict_P,par1_P,par2_P,par3_P,no_P,alpha_dic_P,sigma_dic_P]
= PrecomputeDict1_2(n,savedict,alpha_dic_P,sigma_P)

Computes the different dictionaries for the estimation

INPUTS
Mandatory
n:                      Length of the signal
savedict:               Name of the mat file where the precomputed
                        dictionary is stored
Optional
alpha_dic_P:            Vector of the different fractional orders

sigma_P                 Vector of different standard deviations for the start of
                        the computation of the estimation dictionaries

OUTPUTS
dict_P                  Cell array containing the name of the different
                        estimation dictonaries corresponding to the
                        different sigmas in the vector sigma
par1_P,par2_P,par3_P    Cell arrays: parameters of the dictionaries
no_P                    Vector of the norms of the different
                        dictionaries
alpha_dic_P:            Vector of the fractional orders. All the fractional
                        orders are in each dictionary
sigma_dic_P:            Cell array of the different sigma vectors for each
                        dictionary

See also MakeFracGauss1_2.m, NormFracGauss.m, MakeDic.m, SeisCharac1_4.m
```

11. SeisCharac1_4.m Version 1.4

Delineate and characterize transitions from seismic events

```
function[v_detection,v_alpha] = SeisCharac1_3(data,ratio,whichdic,savedetectest,savedic
trace_nb,par_detection,par_window,par_estimation,par_bcr_d,par_bcr_e,FLAG)

Find the locations and fractional orders of seismic events

INPUTS:
data:              Seismic data (matrix or cube)
ratio:             Ratio between the initial length of the traces and
                   the length of the traces currently analyzed
whichdic:          String: values -> 'detection', 'estimation' or 'both'
                   1- if whichdic = 'detection' -> only computes the
                   detection dictionary
                   2- if whichdic = 'both' -> computes the detection &
                   estimation dictionaries
savedetectest      Name of the detection dictionary
savedict:          Name of the precomputed estimation dictionary
trace_nb:          Vector of length 2 containing the # of the 1st trace
                   to be processed and the # of the last trace to be
                   processed.
                   1- trace_nb(1): # of the 1st trace
                   2- trace_nb(2): # of the last trace
par_detection:     Vector of length 2 containing the parameters for the
                   detection
                   1- par_detection(1): percentage of the norm
                   of the coefficient => par_detection(1)*maxval is the
                   threshold level for the detection
                   2- par_detection(2): minimum width of the sets of
                   coefficients associated to one event
par_window:        Vector of length 2 containing the parameters for the
                   windowing
                   1- par_window(1): # of sigma we want for half the
                   width of the window
                   2- par_window(2): # of sigma we want for the width of
                   the decay of the window
par_estimation:    Vector of length 2 containing the parameters for the
                   estimation (see par_detection)
par_bcr(_d;_e):    Vector of length 3 containing the parameters for the
                   BCR for detection and estimation (_d or _e)
                   1- par_bcr(1): beta
                   2- par_bcr(2): if =1 -> exponential decrease
                                     =0 -> linear decrease
                   3- par_bcr(3): speed1 for Block Solver (BCRDaubechies1_2.m)
```

```
                    4- par_bcr(4): speed2 for Block Solver (BCRDaubechies1_2.m)
FLAG:               = 1 -> only detection performed (default = 0)

OUTPUT:
v_detection:        Vector containing the locations of the events
                    detected in the Detection part
v_alpha:            Vector containing the fractional orders of the
                    different events at their location. v_alpha(k) =
                    alpha1 means that at the location k, the fractional
                    order is alpha1

See also BuildDic1_3.m, BCRDaubechies1_2.m, BP_Interior.m, Fold1_1.m, Window1_3.m,
Estimate1_1.m, TableFracGauss1_2.m
```

12. `whichalpha1_1.m` Version 1.1

    Give the fractional-order of a specific atom in a dictionary made with fractional derivatives of the Gaussian

```
function[alpha_answer] = whichalpha1_1(ind,n,sigma,alpha,loopalpha)

ONLY FOR FRACGAUSS DICTIONARY!
Gives the alpha corresponding to the index ind in the long coefficient
vector in the merged dictionary made of the different sigmas in the
vector sigma and the different alphas in the vector alpha

INPUTS:
ind:                Index in the coefficient vector
n:                  Length of the signal
sigma:              Vector of the different sigmas for the merged dictionary
alpha:              Vector of the different alphas for the merged dictionary
loopalpha:          Flag:
                    = 1 if loop on alpha to build the dictionary is the 1st one
                    = 0 otherwise

OUTPUTS:
alpha_answer:        Alpha corresponding to ind

See also chunk1_1.m, Estimate1_1.m
```

13. whichsigma1_1.m Version 1.1

Give the standard deviation of a specific atom in a dictionary made with fractional derivatives of the Gaussian

```
function[sigma_answer_sample,sigma_answer] = whichsigma1_1(ind,n,sigma,alpha,loopsigma)

ONLY FOR FRACGAUSS DICTIONARY!
Gives the sigma corresponding to the index ind in the long coefficient
vector in the merged dictionary made of the different sigmas in the
vector sigma and the different alphas in the vector alpha

INPUTS:
ind:                  Index in the coefficient vector
n:                    Length of the signal
sigma:                Vector of the different sigmas for the merged dictionary
alpha:                Vector of the different alphas for the merged dictionaryloopsigma
Flag:
                      = 1 of loop on sigma to build the dictionary is the 1st one
                      = 0 otherwise

OUTPUTS:
sigma_answer_sample:  Sigma  in # of samples
sigma_answer:         Sigma corresponding to ind

See also chunk1_1.m, Estimate1_1.m
```

14. `Window1_3.m` Version 1.3

Window out the detected events

```
function[M,sigma_blob_sample,alpha_blob,sigma_blob,v_detection] =
Window1_3(s,cmat,cfold,tolchunk,distblob,alpha_dic,sigma_dic,
loopsigma,table,halfnb_sigma_width,nb_sigma_decay)

Window out the signal s

INPUTS:
s:                      Signal we analyze
cmat:                   Matrix of the coefficient (output of BCRDaubechies)
cfold:                  Vector of the folded coefficient cmat (length(cfold)=n)
tolchunk:               Tolerance for the function chunk (keep the moduli
                         larger than tolchunk)
distblob:               Minimum distance between the chunks allowed (in GetBlob.m)
alpha_dic:              Vector of the fractional orders for the dictionary in
                        which cmat is decomposed in
sigma_dic:              Vector of the frequencies for the dictionary in which
                        cmat is decomposed in
loopsigma:              Flag:
                        = 1 if loop on sigma to build the dictionary is the 1st one
                        = 0 otherwise
table:                  Vector giving the locations in the signal of
                        the coefficient corresponding to a particular
                        atom in FracGauss dictionaries
halfnb_sigma_width:     How many times sigma for half the width of the
                        1 window
nb_sigma_decay:         How many times sigma for the width of the
                        gaussian decay

OUTPUTS:
M:                      Matrix whose columns contain the different
                        windowed signals from s
sigma_blob_sample:      Vector containing the std deviations of the atoms
                        correlating the best with each of the windowed
                        signal (in # of samples)
alpha_blob:             Vector containing the alphas of the atoms
                        correlating the best with each of the windowed
                        signals
sigma_blob:             Vector containing std deviations of the atoms
                        correlating the best with each of the windowed
                        signal
v_detection:            Vector containing the locations of the events
```