# Sparse channels estimation applied in software defined radio

Elaine Crespo Marques

# Sparse Channels Estimation Applied in Software Defined Radio

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Paris

École doctorale n°626 Institut Polytechnique de Paris (IP Paris)
Spécialité de doctorat : Réseaux, Informations et Communications

Thèse présentée et soutenue à Palaiseau, le 20 Décembre 2019, par

**ELAINE CRESPO MARQUES**

Composition du Jury :

| | |
|---|---|
| Jacques-Olivier KLEIN<br>Professeur, C2N, Université Paris-Saclay | Président |
| Hassan ABOUSHADY<br>Maître de Conférences HdR, Sorbonne Université (LIP6) | Rapporteur |
| Raimundo Carlos SILVÉRIO FREIRE<br>Professor Titular, Universidade Federal de Campina Grande (DEE) | Rapporteur |
| Pietro MARIS FERREIRA<br>Maître de Conférences HdR, CentraleSupélec, Université Paris Saclay (GeePs) | Examinateur |
| Lírida NAVINER<br>Professeur, IP Paris (LTCI) | Directrice de thèse |
| Hao CAI<br>Associate Professor, Southeast University (National ASIC System Engineering Center) | Co-encadrant de thèse |

# Acknowledgements

The last years were full of intense emotions, a mix of cries and laughs, moments of hope and despair, exchanging experiences and knowledge. I thank God for my life, for giving me health and enabling incredible people to cross my path. This thesis would not have been possible without the support of several people who have gone through my life before and during it. They rendered the work and moments of sadness less difficult, making this period of my life unforgettable.

It is very difficult to name all the people who contributed to make this day come. The following people are very important to me but not the only ones. I would like you to know that even though you have not been mentioned here I am very grateful for all the help you have given me.

I would like to deeply thank my husband, friend and companion Nilson for all support, love and patience over all these years. Especially during this period of the doctorate. This doctorate would never have started and it would have been impossible to finish it without his help both personally and intellectually. Thank you so much for sharing life with me and supporting me unconditionally.

I want to thank my parents, my sister, my grandmothers, my aunt, and all my family for their constant support, love, and encouragement. They have always been here for me, despite the physical distance. The ocean away did not prevent you from ever being present in my thoughts.

I would like to express my deep gratitude to Professeur Lírida Naviner for giving me the opportunity to pursue my internship in 2009 and now my PhD under her supervision. She has always been much more than a supervisor. Even in busy times, she has always supported me unconditionally. Thank you so much for these years of learning both in the professional and personal fields. Thank you for all the support and trust, for all the help before and during this thesis.

My sincere thanks also to Hao Cai for his support during my thesis. I would also like to express my acknowledgements to all the jury members: Hassan Aboushady, Jacques-Olivier Klein, Pietro

Maris Ferreira, and Raimundo Carlos Silvério Freire for accepting to be part of my PhD committee and for the time and attention dedicated to read this manuscript and their remarks. In particular to Hassan Aboushady and Raimundo Carlos Silvério Freire for agreeing to be the reviewers of this dissertation. Thanks also to Hassan Aboushady and Pietro Maris Ferreira to be part of my *mi-parcours*.

Coming to Télécom Paris and being a PhD. student here was an incredible experience. I have always felt welcome here. Thanks to everyone I met at Télécom Paris as well as everyone who even not knowing personally allows this place to be a place of sharing knowledge and experience of different life and cultures in a pleasant way. Doing my doctorate at Télécom Paris gave me very happy moments, knowledge and new friendships. Thanks to the COMELEC department for welcoming me, especially to the SSH group for the great working environment.

I would like to thank as well Alain Sibille, Florence Besnard, Hamidou, Karim, and Marianna Baziz for making my stay at Télécom Paris so memorable. You helped me so much with bureaucracy and so many other things. Thank you so much Chantal Cadiat and Yvonne Bansimba not only for helping with the bureaucracies but especially for the countless conversations and laughter. Thanks also to the teachers Sophie P. and Véra D. for their great and fun English classes.

Furthermore, many thanks to all the colleagues and friends who shared the office with me and also those I met at Télécom Paris. Sharing my daily routine with you was very rewarding. Thanks for the discussions, lunches, coffee and all the great moments we spent together inside and outside Télécom Paris. Special thanks to Akram, Amir, Bader, Camille P., Daniel, Etienne, Lucas, Fábio, Francisco, Gabriel, Ganesh, Homa, Imene, Juan, Maha, Maarouf, Maurício, Mehrasa, Michele N., Moemoea, Natasha, Patrick, Pedro, Phu, Rafael, Sahar, Soumaya, You, and Xuecan.

Coming to Paris to do my PhD was also an opportunity to meet and live closer with friends I already knew. Thank you so much for your support since the day I arrived and for all the moments we spent together. It was great to live closer with you. In addition, during this period in Paris I have been gifted with new friendship. The moments we spent together will be forever in my memory. In particular, thanks to Adriana, Alexandre, Anderson, Andréa, Anne, Benjamin, Bruno, Camille E., Camille N., Chen, Christine, Éric, Eva, Flávia, Héctor, Jean-François, Julie G., Julie M., Julie T., Léa D., Léa M., Mengqi, Monique, Nardjesse, Pamela, Sara, Sophie N., and Thayane.

Thanks also to my colleagues and friends who stayed in Brazil. Your support before and during the doctorate was very important to me. Thank you for the countless conversations and cheering. In particular my thank you very much to: Anna, Camila, Caio, Ernesto, Jacques, Luciana, Marta,

Michele D., Priscila, Raquel, Renato, and Verdenia. I would also like to thank the support of the Brazilian government and all the people involved.

I will always remember this period that I lived with all of you. I hope to meet you again several times whether in Paris, Brazil or anywhere in the world.

I couldn’t have done this without all of you. Thank you very much!

# Abstract

Communication channels are used to transmit information signals. However, these channels can cause several distortions on the signal to be transmitted, such as attenuation, multipath loss and Doppler shift, among others. For a better message recovery, the receiver can estimate the channel and bring more reliability to the communications systems.

Several communications systems, for example high-definition television, mmWave system, wideband HF and ultra-wideband have sparse channels. This characteristic can be used to improve the performance of the estimator and reduce the size of the training sequence so decreasing the consumption power and bandwidth.

This thesis handles the channel estimation problem by investigating methods that exploit the sparsity of the channel. The study of Compressive Sensing and its sparse recovery algorithms led to the proposition of a new algorithm called Matching Pursuit based on Least Square (MPLS).

The use of neural networks (NN) to sparse signals estimation was also explored. The work focused on NN inspired by sparse recovery algorithms such as Learned Iterative Shrinkage-Thresholding Algorithm (LISTA). This resulted in two approaches that improve LISTA performance as well as to a new neural network suitable to estimate sparse signals.

# Table of Contents

# List of Figures

# List of Tables

# List of Acronyms

|  |  |
|---|---|
| **ADC** | Analog to Digital Converter. |
| **AIC** | Analog to Information Converter. |
| **AMP** | Approximate Message Passing. |
| **AST** | Affine Scaling Transformation. |
| **AWGN** | Additive White Gaussian Noise. |
| **BAOMP** | Back-tracking based Adaptive Orthogonal Matching Pursuit. |
| **BCS** | Bayesian Compressive Sensing. |
| **BP** | Basis Pursuit. |
| **BPDN** | BP de-noising. |
| **CIR** | Channel Impulse Response. |
| **CNN** | Convolutional Neural Network. |
| **CoSaMP** | Compressive Sampling Matching Pursuit. |
| **CP** | Chaining Pursuit. |
| **CR** | Cognitive Radio. |
| **CS** | Compressive Sensing. |
| **CSI** | Channel State Information. |
| **D-AMP** | Denoising-based AMP. |

**DnCNN** Denoising Convolutional Neural Network.

**DNN** Deep Neural Network.

**DNN-WISTA** Deep Neural Network-structured WISTA.

**DNN-IHTA** Deep Neural Network-structured IHTA.

**D-OMP** Differential Orthogonal Matching Pursuit.

**D$l_0$RE** Deep $l_0$-Regularized Enconder.

**DOA** Direction-of-Arrival.

**DPP** Delay Power Profile.

**DS** Dantzig Selector.

**ECG** Electrocardiogram.

**EM** Expectation-Maximization.

**FBP** Forward-Backward Pursuit.

**FISTA** Fast Iterative Shrinkage Thresholding Algorithm.

**FOCUSS** Focal Underdetermined System Solution.

**FPGA** Field Programmable Gate Array.

**GBP** Greedy Basis Pursuit.

**GOAMP** Generalized Orthogonal Adaptive Matching Pursuit.

**GOMP** Generalized Orthogonal Matching Pursuit.

**GP** Gradient Pursuit.

**GPP** General Purpose Processors.

**GraDeS** Gradient Descent with Sparsification.

**HDTV** High-Definition Television.

**HELU** Hard thrEsholding Linear Unit.

**HF** High Frequency.

**HHS** Heavy Hitters on Steroids.

**IBTMC** Inter-Burst Translational Motion Compensation.

**IHT** Iterative Hard Thresholding.

**IHTA** Iterative Half Thresholding Algorithm.

**IoT** Internet of Things.

**IRLS** Iterative Reweighted Least Squares.

**ISAR** Inverse Synthetic Aperture Radar.

**ISDB-T** Integrated Services Digital Broadcasting-Terrestrial.

**ISTA** Iterative Soft Thresholding.

**LAMP** Learned Approximate Message Passing.

**LARS** Least Angle Regression.

**LASSO** Least Absolute Shrinkage and Selection Operator.

**LDAMP** Learned Denoising-based Approximate Message Passing.

**LISTA** Learned Iterative Shrinkage-Thresholding Algorithm.

**LS** Least Square.

**LVAMP** Learned Vector Approximate Message Passing.

**MAP** Maximum a posteriori.

**MIMO** Multiple-Input Multiple-Output.

**ML** Maximum Likelihood.

**MMP** Multipath Matching Pursuit.

**MMSE** Minimum Mean Squared Error.

**MP** Matching Pursuit.

**MPLS** Matching Pursuit based on Least Squares.

**MRI** Magnetic Resonance Imaging.

**MSD** Mean-Square Deviation.

**NMSE** Normalized Mean Squared Error.

**NN** Neural Network.

**NVIS** Near Vertical Incidence Skywave.

**OAMP** Orthogonal Approximate Message Passing.

**OLS** Oracle Least Square.

**OMP** Orthogonal Matching Pursuit.

**OTH** Over-the-Horizon.

**PAM** Pulse Amplitude Modulation.

**PNN** Proposed Neural Network.

**PSNR** Peak Signal to Noise Ratio.

**QPSK** Quadrature Phase Shift Keying.

**ReLU** Rectified Linear Unit.

**RF** Radio Frequency.

**RIC** Restricted Isometry Constant.

**RIP** Restricted Isometry Property.

**RNN** Recurrent Neural Network.

**ROMP** Regularized OMP.

**RVM** Relevance Vector Machine.

**SBL** Sparse Bayesian Learning.

**SCA** Software Communications Architecture.

**SDR** Software Defined Radio.

**SGD** Stochastic Gradient Descent.

**SGP** Stochastic Gradient Pursuit.

**SLSMP** Sequential Least Squares Matching Pursuit.

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

**SoC** System on Chip.

**SP** Subspace Pursuit.

**SpAdOMP** Sparse Adaptive Orthogonal Matching Pursuit.

**SpaRSA** Sparse Reconstruction by Separable Approximation.

**StOMP** Stagewise Orthogonal Matching Pursuit.

**SURE-AMP** Stein's Unbiased Risk Estimate Approximate Message Passing.

**SURE-TISTA** Stein's Unbiased Risk Estimate based-Trainable Iterative Thresholding Algorithm.

**TISTA** Trainable ISTA.

**TOMP** Tree-based Orthogonal Matching Pursuit.

**TSMP** Tree Search Matching Pursuit.

**VAMP** Vector Approximate Message Passing.

**WISTA** Weighted Iterative Shrinkage Thresholding Algorithm.

**WSN** Wireless Sensors Network.

# List of Notation

| | |
|---|---|
| $(\cdot)^H$ | hermitian transpose |
| $(\cdot)^T$ | transpose |
| $(\cdot)^\dagger$ | pseudo-inverse |
| $(\cdot)^{-1}$ | inverse |
| $\mu(\cdot)$ | coherence |
| $\|\|\cdot\|\|_p$ | $l_p$ norm - $\|\|\mathbf{x}\|\|_p = (\sum_{i=1}^{n} \|\mathbf{x}_i\|^p)^{1/p}$ |
| $x(i)$ | $i^{th}$ element of the vector $\mathbf{x}$ |
| $\mathbf{x}_i$ | vector $\mathbf{x}$ at iteration/layer $i$ |
| $\mathbf{A}(\Lambda_i)$ | submatrix of $\mathbf{A}$ containing only those columns of $\mathbf{A}$ with indices in $\Lambda_i$ |
| $\mathbf{h}$ | N $\times$ 1 sparse signal vector to be estimated |
| $\hat{\mathbf{h}}$ | estimate of $\mathbf{h}$ |
| $\mathbf{y}$ | M $\times$ 1 received signal vector |
| $\mathbf{n}$ | M $\times$ 1 noise vector |
| $\mathbf{A}$ | M $\times$ N measurement matrix |
| $\Psi$ | N $\times$ N basis matrix |
| $N$ | length of the sparse signal $\mathbf{h}$ |
| $M$ | length of the received signal $\mathbf{y}$ |
| $N_L$ | the number of layers of the neural network |
| $s$ | signal‘s sparsity |
| $\Lambda_i$ | set of the indices chosen untill iteration $i$ |
| $\delta_s$ | the smallest number that achieves RIP |
| $N_{it}$ | number of iterations |

# Chapter 1

# Introduction

This chapter introduces the reader to the context of this thesis and outlines the motivation and contributions of the present work.

## 1.1 Motivations

In the last decades, the development of technological solutions in communications has given rise to a new radio concept called Software Defined Radio (SDR) [1]. It is an emerging technology in which previously hardware-based features have become software-defined, enabling users to enter new applications when using an SDR. It presents several advantages for the development of wireless solutions in civil and military communications such as interoperability between troops of different forces and nations, the portability of waveforms, and the possibility of updating with the latest advances in radio communications without requiring a new hardware.

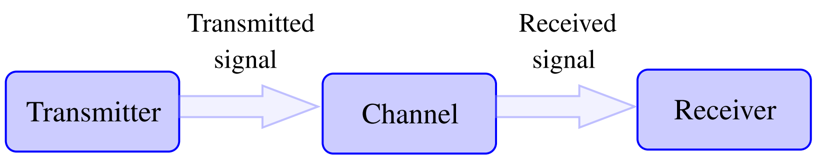


Figure 1.1 – Elements of a communication system.

A communication system is basically composed of three elements: transmitter, channel and receiver as shown in Fig. 1.1 [2]. Communications channels can cause several distortions on the signal, such as attenuation, multipath loss and Doppler shift, among others [2]. Consequently, to better recover the message sent by the transmitter, an estimate of the channel can be used to compensate these effects, resulting in an accurate signal demodulation, equalization, and decoding.

Indeed, high-quality channel estimation is an essential feature of reliable communication systems.

One of the major challenges of communication systems is to provide accurate channel state information (CSI) at the receiver [3]. With the estimated CSI, transmitted symbols can be recovered at the receiver. In order to do this, several channel estimation techniques to provide CSI have been developed. In general, these techniques can be categorized into three classes as illustrated in Fig. 1.2 [4].

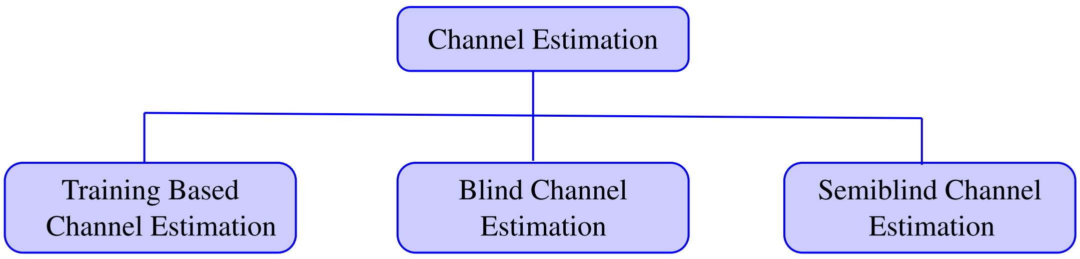

Channel Estimation
Training Based Channel Estimation
Blind Channel Estimation
Semiblind Channel Estimation

Figure 1.2 – Classification of channel estimation techniques.

The training based channel estimation uses some known symbols, the training sequence, sent before and/or during the communication to estimate the channel from the measured outputs to training sequence. Fig. 1.3 illustrates the transmitted symbol vector with $T$ training and $D$ data symbols. However, these known signals consume power and bandwidth impacting on the channel spectral efficiency due to the use of communication resources to transmit these symbols instead of transmitting data. Moreover, in time-varying channels, the training sequence needs to be transmitted periodically, resulting in further loss of channel throughput. Therefore, it is interesting to reduce the required amount of the training symbols while keeping a sufficient estimation accuracy. To achieve this, the characteristic of the channel can be used to improve the performance of its estimator.

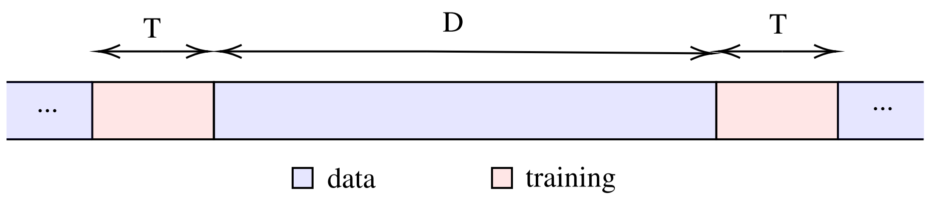


Figure 1.3 – Transmitted symbol vector with $T$ training and $D$ data symbols.

On the other hand, blind techniques focus on statistical or deterministic properties of the system. They do not exploit the knowledge of the training sequence. The main advantage of the blind channel estimation techniques is the possibility of eliminating the training sequences. Nonetheless, in general, these techniques require long data record leading to a slow convergence rate [3].

Finally, the semiblind channel estimation algorithm is a combination of training based channel estimation and blind channel estimation. It utilizes the known training symbols and the unknown data symbols to perform channel estimation.

A communication channel is usually modeled by its channel impulse response (CIR), which is a vector whose elements represent the complex gains associated with each multipath component of the channel. In various communication systems, such as high-definition television (HDTV) [5, 6], mmWave system [7, 8], wideband HF [9, 10], ultra-wideband [11–13], massive MIMO [14] and underwater communication systems [15, 16], the channels can be considered sparse channels. In other words, their impulse responses are characterized by a few significant terms that are widely separated in some domain, that is, many coefficients are close to or equal to zero. Indeed, in wireless communication systems, for example, several components of the signal arrive at the receiver with a delay due to the multipath caused by the environment, making its response sparse [17].

The length of a sampled sparse channel can reach hundreds of symbol intervals, although the majority of taps in the sampled channel are near zero-valued. The channel sparsity is defined as the number of non-zero taps of the channel [18, 19]. The sparsity can significantly reduce the effective signal space dimension. That is, the size of the possible signals set in a sparse channel can be much lower than the original (supposing non-sparse) signal space dimension.

Traditional channel estimation techniques such as least squares (LS) do not exploit the channels sparsity resulting in over-parameterization and poor performance of the sparse channel estimator [2]. Furthermore, classical estimation algorithms become too complex for tackling these channels [20].

The sparsity characteristic of the signal can be used to reduce the cost and complexity of the signal estimation. For example, using compressive sensing (CS), these signals can be reconstructed from fewer measurements than the required by the Shannon-Nyquist sampling theorem [18, 21–24]. The great advantage of CS is that it allows you to digitize only the relevant signal information at a much lower sample rate than the Nyquist rate, so the digitized signal is already in a compressed form. Compressive sensing has been applied to sparse channel estimation [8, 15, 18, 20, 25–27]. The structure of sparse channels can be exploited using sparse reconstruction algorithms such as Matching Pursuit (MP) [28], Basis Pursuit (BP) [29] and others [30] leading to better estimation performance.

The channel estimation can be formulated as an optimization problem. An optimization problem can be solved by using numerical algorithms that iteratively refine their solution. Generally, in channel estimation, an accurate solution should be found with a small number of iterations. In addition, these algorithms require some parameters adjustments that if they are not well chosen may decrease the algorithm performance.

In order to overcome some issues of these algorithms, model-driven neural networks (NNs) are becoming popular in communications systems [31–42]. Some of them are based on algorithms which have performance guarantees and NNs tools, combining the best of both.

Starting with the study of compressive sensing, sparse recovery algorithms, and neural network, this thesis aims at contributing to the development of algorithms suitable for sparse channel estimation. In this work, the channel estimation is formulated as a signal recovery problem. The developed approaches are intended to be used in Software Defined Radio (SDR) applications. However, they can be applied in other areas where the signal of interest can be considered sparse or compressible.

## 1.2 Thesis Contributions

This thesis is focused on sparse channel estimation. The main research contributions are:

- Study of sparse recovery algorithms (see Chapter 3) resulting in the publication of a review article in IEEE Access [30].
- Performance comparison between some sparse recovery algorithms that have been presented in the literature (see Section 3.4).
- Proposition of a greedy algorithm called Matching Pursuit based on Least Squares (MPLS) (see Section 5.1).
- Wideband HF channel estimation based on CS (see Section 5.2).
- Proposal of a new approach to improve Learned Iterative Shrinkage-Thresholding Algorithm (LISTA) estimation performance (see Section 5.3).
- Proposal of a new neural network to sparse signal estimation (see Section 5.4).
- Analysis of alternative shrinkage functions to be used in LISTA (see Section 5.5).

## 1.3 Organization of the Thesis

This report is organized into six chapters, divided as follows:

**Chapter 2** is dedicated to introduce the background of this thesis. First, Software Defined Radio is addressed. Then, the concepts of sparsity and sparse signal are discussed. Finally, the basis of compressive sensing are presented and some application areas are illustrated.

An overview of sparse recovery algorithms is presented in **Chapter 3**. These algorithms can be classified into three categories: convex relaxation, non-convex optimization techniques and greedy algorithms. Some algorithms of each category are addressed in this chapter. Moreover, performance comparisons of them are also analyzed.

**Chapter 4** discusses some neural networks proposed in the literature to deal with sparse signals estimation.

**Chapter 5** presents the main contributions of this thesis. Based on the analysis made in Chapter 3, a sparse recovery algorithm called Matching Pursuit based Least Square (MPLS) is proposed. This algorithm as well as other greedy algorithms are applied to Wideband HF channel estimation. In addition, the study of the neural networks addressed in Chapter 4 led to the suggestion of two approaches to improve LISTA estimation performance. Furthermore, a neural network is proposed to sparse signal estimation. Finally, a performance comparison between the techniques proposed in this thesis and other sparse recovery algorithms is analyzed.

**Chapter 6** concludes the work accomplished during this thesis and presents some perspectives of future research directions.

# Chapter 2

# Basic Concepts

This chapter introduces some concepts related to the thesis subject. Firstly, some characteristics of Software Defined Radio are addressed. Then, sparse signal and sparsity are explained. Finally, key concepts of the compressive sensing theory are presented and its applications are illustrated.

## 2.1 Software Defined Radio (SDR)

A radio is a device that wirelessly transmits or receives signals in the radio frequency (RF) part of the electromagnetic spectrum. It can be found in many everyday items such as cell phones, television, and vehicles. With the amount of data and ways of communication between people growing very fast, modifying radio devices efficiently to respond to today's different communication needs has become a major challenge and goal of civil and military circles.

Using traditional hardware based radio devices, a physical intervention is required to modify them limiting cross-functionality and higher production costs. In contrast, Software Defined Radio (SDR) represents a great alternative providing an efficient and comparatively inexpensive solution to this problem [43].

The SDR Forum defines an SDR as "a radio in which some or all of the physical layer functions are software defined" [43]. In other words, an SDR is a radio implemented with generic hardware that can be programmed to transmit and to receive a variety of waveforms. A Waveform can be defined as the set of transformations applied to information to be transmitted and the corresponding set of transformations to convert received signals back to their information content [44].

This progress is due to improvements in several areas, such as embedded systems, analog to digital converters (ADC), digital transmission, digital signal processing, multi-band antennas, software architectures, and the ability to run new general purpose processors (GPP). Based on this, SDR anticipates important advantages for the development of wireless solutions in civil and military communications systems. Among the features envisioned are the interoperability between troops of different military forces and nations, the portability of waveforms, and the possibility of updating with new wireless features and capabilities without requiring new hardware, reducing logistical support and operating expenditures [43]. SDR also allows that new products be more quickly introduced into the market using a common platform architecture and reducing development costs. In addition, SDR is viewed as the most suitable platform for the development of cognitive radios (CR).

The high-level functional model of an SDR consists of an RF front-end subsystem that performs channel selection, down-conversion to the basic band, and data routing to a processing unit based on software. In this unit, the associated digital data set is presented to several layers (e.g., link modules, network, security) to perform decoding tasks suitable for extracting the desired information. This process is inverted on the transmission side, where the input signal is encoded and modulated, appropriately for the transmission of information. This signal is then passed to the RF subsystem for insertion into the radio channel.

Due to the multiplicity of concepts related to the functional model described in the previous paragraph, several efforts have been made to standardize key elements within the SDR architecture, providing a common platform for the development of radio equipment. Supported standards can be proprietary or industry-standard, being developed through a consensus process. While the first approach brings product differentiation to manufacturers, the second strategy commoditizes the technology, enabling third party support in building the radio platform to achieve specific business objectives. One of the most typical zones for standardization is the application framework, which provides a common software operating environment in which the vendor freely provides a set of interfaces for installing, configuring, controlling, and releasing application operation on a SDR platform. Examples of application frameworks relevant to SDR systems include the Open Mobile Alliance and the Software Communications Architecture (SCA) [45].

## 2.2 Sparse Signals

Sparse signals are characterized by the concentration of large part of its energy in a small fraction of its duration. That is, few of their coefficients have non-zero values, and that concentrate the relevant information. These signals can be found in several domains [46]. For example, most of the images and the audio signals have a sparse decomposition on a base of wavelets and time-frequencies, respectively. The next paragraphs introduce sparse signals based on their vector representation.

Consider the support of a vector $\mathbf{x}$ is defined as the index set of its non-zero elements, that is:

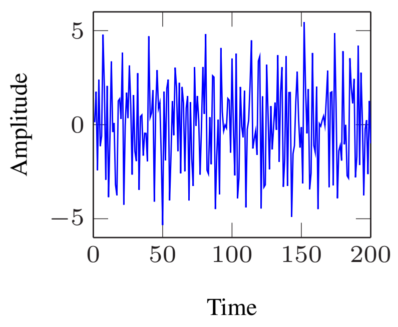
$$supp(\mathbf{x}) = \{i : x(i) \neq 0\} \tag{2.1}$$

The sparsity of a signal $\mathbf{x}$ can be measured by its $l_0$-norm (see (2.2)). A $s$-sparse signal has no more than $s$ non-zero coefficients. On the other hand, let $I(k)$ be the $k^{th}$ largest component of $\mathbf{x}$ sorted by magnitude from largest to smallest, $r \geq 1$, and $C$ a constant, the coefficients $x(i)$ of a compressible signal decrease in magnitude according to (2.3) [47]. Due to their rapid decay, such signals can be well approximated by $s$-sparse signals, keeping just the $s$ largest coefficients of $\mathbf{x}$.

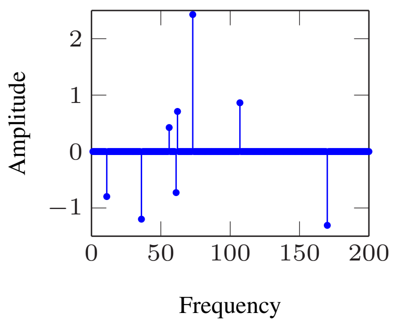
$$||\mathbf{x}||_0 = \#supp(\mathbf{x}) = \#\{i : x(i) \neq 0\} \tag{2.2}$$

$$|x(I(k))| \leqslant Ck^r \quad k = 1, \ldots, n \tag{2.3}$$

Fig. 2.1 shows a 200 samples length time-domain signal (Fig. 2.1a) representing 8 distinct sinusoids (Fig. 2.1b). This figure is an example of 8-sparse signal in frequency domain, that is, it can be seen in Fig. 2.1b that only 8 non-zero values exist among the 200 frequencies.



(a) Time domain



(b) Frequency domain

Figure 2.1 – Samples of 8 sinusoids in (a) time and (b) frequency domains.

Dealing with sparse signals presents several advantages such as lower computational complexity, lower processing time and less memory required during vector and/or matrix multiplication.

Therefore, the performance of sparse signal processing algorithms can be improved if the sparsity characteristic of the signal is taken into account.

## 2.3 Compressive Sensing (CS)

Nyquist–Shannon sampling theorem establishes a condition for a sample rate that permits perfectly reconstructing a signal of finite bandwidth, i.e., the sample rate (Nyquist rate) has to be twice the bandwidth of the signal. Traditionally, signal acquisition and transmission firstly sample the signal at the Nyquist rate, then compress it. Nevertheless, it is worth to mention that in several scenarios, the Nyquist rate can be very high, resulting in too many samples, which makes it difficult to process it. In other words, it leads to many challenges related to high sampling rates for data acquisition and large amounts of data for storage and transmission.

The compressive sensing (CS) theory, also known as compressed sensing or compressive sampling, appears as an alternative approach to the traditionally signal processing. In order to reduce energy consumption and time processing, improve storage capacities and facilitate signal processing, CS performs signal sampling and compression simultaneously [21, 22, 47–49]. Employing sparse representation, that is, the underlying assumption that the signal is sparse or compressible by some transforms (e.g., Fourier, wavelets), CS allows recover signals from fewer measurements than the Nyquist rate (sub-Nyquist sampling) reducing data acquisition costs.

Although the term “compressive sensing” has become popular in the last decades, its idea is very old. Gaspard Riche de Prony can be considered as one of the first to deal with ideas related to sparse estimation [50]. In 1795, he was interested in recovering small number of exponential terms sampled in the presence of noise [51]. On the other hand, works on Geology and Geophysics in the eighties showed that wideband seismic signal can be recovery by very incomplete measurements [52, 53] and one of the first paper to explicit use of $l_1$-norm for signal reconstruction was written [54]. The general algorithmic principles that are used in compressive sensing were published in [28, 55, 56]. Nevertheless, the foundation of compressive sensing is related to [21, 22, 47–49].

### 2.3.1 Basics on Compressive Sensing

Compressive sensing allows to achieve two highly targeted objectives [22]:

- to reduce the energy for transmission and storage through the projection of the information into a lower dimensional space;

- to reduce the power consumption by reducing the sampling rate to the signal's information content rather than to its bandwidth.

Fig. 2.2 illustrates the three main steps involved in compressive sensing [23, 57]:

- Sparse Representation;
- CS acquisition (measurement);
- CS reconstruction (sparse recovery).

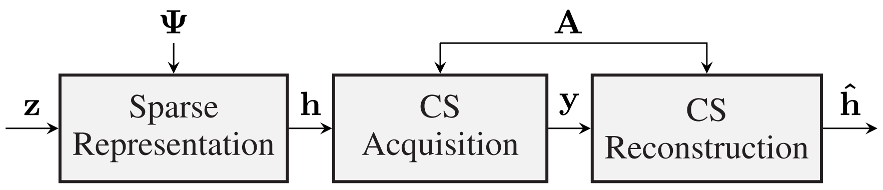


Figure 2.2 – Compressive sensing main steps.

In the first step (Sparse Representation), the signal is represented as a projection on a suitable basis, i.e., a linear combination of only $s$ basis vectors, with $s \ll N$. It means that a signal $\mathbf{z}$ with $N \times 1$ column vector in its original representation can be represented with a basis of $N \times 1$ vectors $\{\psi_i\}_{i=1}^{N}$. Let $\Psi$ be the $N \times N$ basis matrix, the signal can be represented in its sparse form $\mathbf{h}$ by:

$$\mathbf{z} = \Psi\mathbf{h} \tag{2.4}$$

In the second step (CS Acquisition), the signal $\mathbf{z}$ is measured by sampling it according to a matrix $\Phi \in \mathbf{C}^{M \times N}$, where $\phi_i$ denotes the $i^{th}$ column of the matrix $\Phi$. The system model is defined by:

$$\mathbf{y} = \Phi\mathbf{z} + \mathbf{n} = \Phi\Psi\mathbf{h} + \mathbf{n} = \mathbf{A}\mathbf{h} + \mathbf{n} \tag{2.5}$$

where $\mathbf{y} = [y_1, y_2, ..., y_M]^T$ denotes the received signal, $\mathbf{h} = [h_1, h_2, ..., h_N]^T$ is the sparse signal vector with $N > M$ and $\mathbf{n}$ is the noise.

The third step (CS Reconstruction) deals with recovery, which is possible if the following two fundamental premises underlying CS are attended [22]:

- *Sparsity* - means that the signal could be characterized by few significant terms in some domain;
- *Incoherence* - states that distances between sparse signals are approximately conserved as distances between their respective measurements generated by the sampling process.

The largest correlation between any two elements of $\Psi$ and $\Phi$ is measured by the coherence between these matrices and it is defined by:

$$\mu(\Phi,\Psi) = \sqrt{N} \max_{1 \leq k,j \leq N} \left| < \varphi_k, \psi_j > \right| \tag{2.6}$$

If $\Phi$ and $\Psi$ contain correlated elements, the coherence is large. On the contrary, the coherence is small. Compressive sensing is mainly concerned with low coherence pairs. In [58], considering $C$ as a constant, the authors showed that if (2.7) holds, then with overwhelming probability one sparse recovery algorithm will recover the signal.

$$M \geq C\mu^2(\Phi,\Psi) s \log N \tag{2.7}$$

Equation (2.7) shows that fewer measurements will be required to recover the signal if the coherence between $\Psi$ and $\Phi$ is small [59].

As illustrated in Fig.2.2, the last part (sparse recovery - CS Reconstruction) recovers the sparse signal from a small set of measurements $\mathbf{y}$ through a specific sparse recovery algorithm [57]. This step concerns the development of efficient sparse recovery algorithms. Some of them are addressed in Chapter 3.

Fig. 2.3 illustrates the relationship between the variables in a noiseless scenario. This work considers a noisy scenario and that the signal to be estimated is already in its sparse representation $\mathbf{h}$. Therefore, the system is defined by:

$$\mathbf{y} = \mathbf{A}\mathbf{h} + \mathbf{n} \tag{2.8}$$

One of the challenges associated with the sparse signal estimation is to identify the locations of the non-zero signal components. In other words, this is finding the subspace generated by no more than $s$ columns of the matrix $\mathbf{A}$, related to the received signal $\mathbf{y}$. After finding these positions, the non-zero coefficients can be calculated by applying the pseudoinversion process.

CS theory addresses two main challenges:

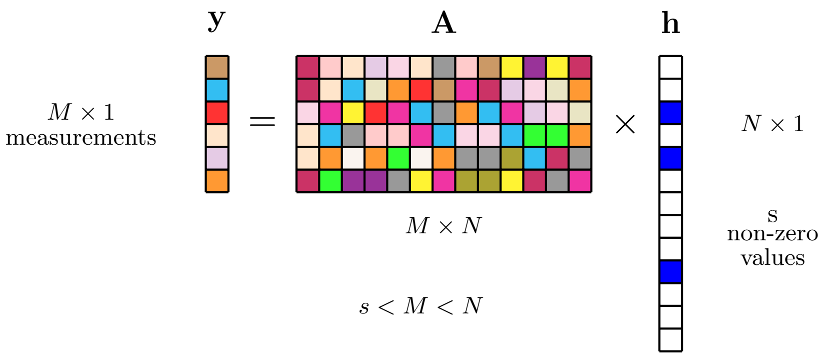


Figure 2.3 – Representation of measurements used in compressive sensing.

- Design of the measurement matrix **A**;
- Development of a sparse recovery algorithm for the efficient estimation of **h**, given only **y** and **A**.

In the first challenge, the goal is to design a measurement matrix **A** which assures that the main information of any $s$-sparse or compressible signal is in this matrix [59]. The ideal goal is to design an appropriate measurement matrix with $M \approx s$.

The measurement matrix is very important in the process of sparse signal recovering. According to [22], if the Restricted Isometry Property (RIP) defined in (2.9) is satisfied, using some recovery algorithm, it is possible to obtain an accurate estimation of the sparse signal **h**, for example solving an $l_p$-norm problem [60]. $\delta_s \in (0,1)$ is the RIC (Restricted Isometry Constant) value and corresponds to the smallest number that achieves (2.9). The smaller $\delta_s$, the closer any submatrix of **A** composed by $s$ columns is to being orthogonal.

$$(1-\delta_s)||\mathbf{h}||_2^2 \leq ||\mathbf{Ah}||_2^2 \leq (1+\delta_s)||\mathbf{h}||_2^2 \tag{2.9}$$

Table 2.1 reproduces a comparison between deterministic sensing and random sensing for the measurement matrix **A** presented in [59]. The random matrices are one approach to obtain a measurement matrix **A** that obeys the RIP condition. Many works deal with random measurement matrices generated by identical and independent distributions (i.i.d.) such as Bernoulli, Gaussian, and random Fourier ensembles [48, 61–63]. However, these matrices require significant space for storage and they have excessive complexity in reconstruction [59]. Furthermore, it is difficult to verify whether these matrices satisfy the RIP property with a small RIC value [59].

Therefore, deterministic matrices have been studied to be used as measurement matrices.

Table 2.1 – Comparison between random and deterministic sensing.

| Random Sensing | Deterministic Sensing |
|---|---|
| Outside the mainstream of signal processing: worst case signal processing | Aligned with the mainstream of signal processing: average case signal processing |
| Less efficient recovery time | More efficient recovery time |
| No explicit constructions | Explicit constructions |
| Larger storage | Efficient storage |
| Looser recovery bounds | Tighter recovery bounds |

In [64] and [65], the authors propose deterministic measurement matrices based on coherence and based on RIP, respectively. Moreover, deterministic measurement matrices are constructed via algebraic curves over finite fields in [66]. Furthermore, a survey on deterministic measurement matrices for CS can be found in [67].

Having defined the appropriate measurement matrix $\mathbf{A}$, $\mathbf{h}$ can be estimated by the least squares (LS) solution of (2.8), i.e., solving the problem (2.10), where $\varepsilon$ is a predefined error tolerance.

$$\min ||\hat{\mathbf{h}}||_2 \ \ subject \ \ to \ \ ||\mathbf{y}-\mathbf{A}\hat{\mathbf{h}}||_2^2 < \varepsilon \tag{2.10}$$

This system is "underdetermined" (the matrix $\mathbf{A}$ has more columns than rows). Normally, an underdetermined system has an infinite number solutions. However, considering that the signal is sparse, the number of solutions is reduced.

Let $\mathbf{A}^\dagger$ be the pseudo-inverse matrix of $\mathbf{A}$ and $\mathbf{A}\mathbf{A}^H$ has an inverse matrix, according to the LS algorithm, the unique solution $\hat{\mathbf{h}}$ of the optimization problem (2.10) is given by (2.11) [2].

$$\hat{\mathbf{h}}_{LS} = \mathbf{A}^\dagger\mathbf{y} = \mathbf{A}^H(\mathbf{A}\mathbf{A}^H)^{-1}\mathbf{y} \tag{2.11}$$

Fig. 2.4 illustrates an example of the estimation of the sparse signal $\mathbf{h}$ using the LS algorithm. It is worth noting that the least squares minimization problem leads to the lowest energy solution, however it may not return a sparse vector (see the estimate $\hat{\mathbf{h}}_{LS}$ in Fig. 2.4). Therefore, alternatives have been sought. By focusing on the sparsity constraint on the solution and solving the $l_0$ norm minimization described by (2.12), it is possible to obtain a sparse approximation $\hat{\mathbf{h}}$. The Lemma 1.2 of [49] shows that if the matrix $\mathbf{A}$ obeys the RIP condition with constant $\delta_{2s} < 1$, (2.12) has an unique solution and $\mathbf{h}$ can be reconstructed exactly from $\mathbf{y}$ and $\mathbf{A}$.

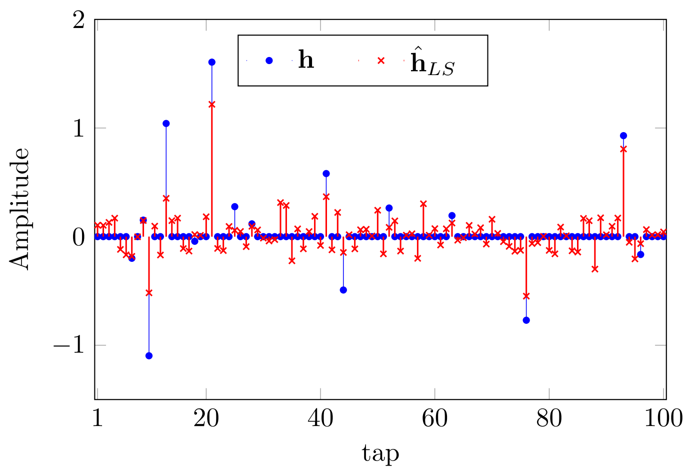


Figure 2.4 – Sparse signal estimation using the LS algorithm.

$$\min ||\hat{\mathbf{h}}||_0 \quad subject\ to \quad ||\mathbf{y} - \mathbf{A}\hat{\mathbf{h}}||_2^2 < \varepsilon \tag{2.12}$$

Unfortunately, an exhaustive search over all $\binom{N}{s}$ possible sparse combinations is required in the $l_0$ minimization problem, which is computationally intractable for some practical applications. Thus, although this gives the desired solution, in practice it is not feasible to solve this equation. The excessive complexity of such a formulation can be avoided with the minimization of the $l_1$ problem (2.13), which can efficiently compute (2.12) under certain conditions, as demonstrated in [68].

$$\min ||\hat{\mathbf{h}}||_1 \quad subject\ to \quad ||\mathbf{y} - \mathbf{A}\hat{\mathbf{h}}||_2^2 < \varepsilon \tag{2.13}$$

One of the advantages of the $l_1$ norm minimization approach is that it can be solved efficiently by linear programming techniques [69]. Moreover, in [70], the authors state that sparse signals can be recovered through $l_1$ minimization if $M \approx 2s\log(N)$.

### 2.3.2 Application of Compressive Sensing

This section overviews some application areas where CS can be suitable [30]. Fig. 2.5 enumerates the areas addressed below.

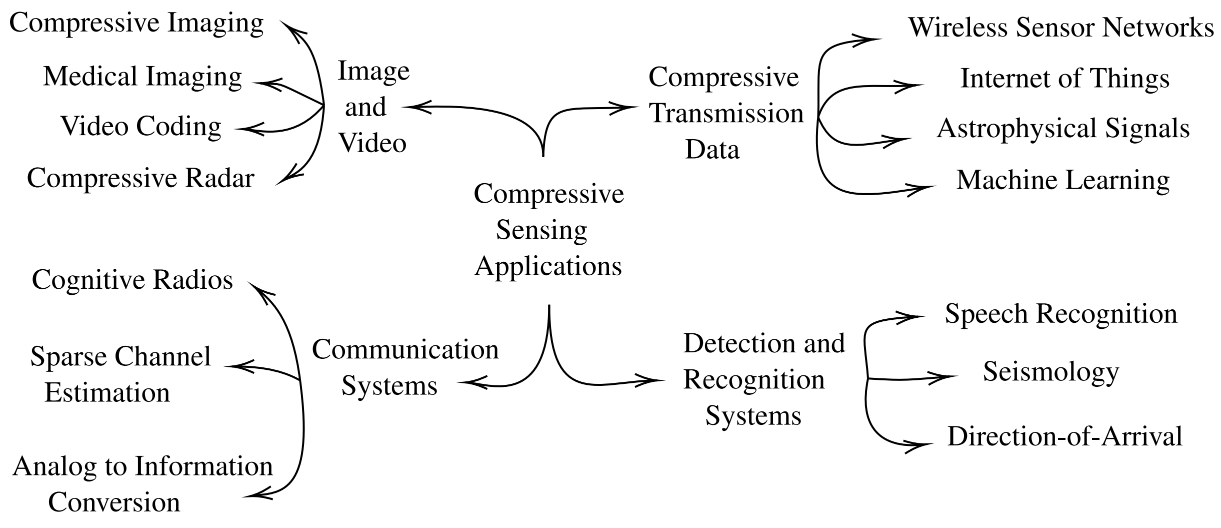


Figure 2.5 – Compressive sensing application areas.

#### 2.3.2.1 Image and Video

**Compressive Imaging** Natural images can be sparsely represented in wavelet domains, so the required number of measurements in compressive imaging can be reduced using CS [71, 72]. One example of application is the single-pixel camera that allows reconstructing an image in a sub-Nyquist image acquisition, that is, from fewer measurements than the number of reconstructed pixels [73].

**Medical Imaging** CS can be very useful for medical imaging. For example, the magnetic resonance imaging (MRI) is a time-consuming and costly process. CS allows to decrease the number of samples, and then to reduce the time of acquisition [74]. Similarly, bio-signals such as ECG signals are sparse in either wavelet or Fourier domain [75]. CS allows to take advantage of the sparsity and reduces the required number of collected measurements [74–77]. A hardware implementation on a system on chip (SoC) platform of a solution to tackle big data transmission and privacy issues is presented in [78].

**Video Coding** Due to the development and the increase of video surveillance, mobile video, and wireless camera sensor networks, wireless video broadcasting is becoming more popular and finding several real-time applications [79, 80]. In these cases, a single video stream is simultaneously transmitted to several receivers with different channel conditions [80]. In order to do this, many new video codecs have been proposed using compressive sensing [80–83].

**Compressive Radar** Radar imaging systems aim to determine the direction, altitude, and speed of fixed and moving objects [84]. By solving an inverse problem using the compressive sensing theory, the received radar signal can be recovered from fewer measurements [84]. Therefore, the cost and the complexity of the hardware of the receiver are extremely reduced [84, 85]. Moreover, the CS has been a novel way to deal with the Inter-Burst Translational Motion Compensation (IBTMC) to achieve the exact recovery of Inverse Synthetic Aperture Radar (ISAR) images from limited measurements [86].

#### 2.3.2.2 Compressive Transmission Data

**Wireless Sensor Networks** Wireless sensor networks (WSNs) require high communication costs and energy consumption. Due to critically resource constraints as limited power supply, communication bandwidth, memory, and processing performance, CS can be used to reduce the number of bits to be transmitted or to represent the sensed data in WSNs [87–90].

**Internet of Things** The use of internet of things (IoT) devices has increased and it is estimated that it will continue to do so in the following years. This includes home automation/control devices, security cameras, mobile phones, and sensing devices [91]. However, the IoT devices have computation, energy, and congestion constraints. Even if they need to transmit large amounts of data, they usually have limited power and low-computation capabilities. Moreover, given the large number of devices connected, they can suffer from congestion and packet drops [91]. Thus, special data transmission strategies have to be developed to enable low-power and low-cost signal processing operations, and energy-efficient communications [91]. Multimedia data usually possesses sparse structures. Therefore, the CS theory emerges as a good strategy to reduce the amount of data that the IoT devices need to transmit with a high fidelity recovery data [92].

**Astrophysical signals** Radio receivers located in outer space suffer from strong restrictions on storage capacity, energy consumption, and transmission rate. To overcome these challenges, sampling architectures using CS provide a data acquisition technique with fewer measurements. Thus, the amount of collected data to be downloaded to Earth and the energy consumption are reduced. The simple coding process with low computational cost provided by the CS promotes its use in real-time applications often found onboard spacecrafts. Moreover, the reconstruction of the signals will be done on Earth where there are much more computing and energy resources than onboard a satellite [93, 94].

**Machine Learning** Machine learning algorithms perform pattern recognition (e.g., classification) on data that is too complex to model analytically to solve high-dimensional problems. However, the amount of information generated by acquisition devices is always huge and ever-growing. It can achieve gigabytes of data or more that exceeds the processing capacity of the most sophisticated machine learning algorithms [95]. To reduce the energy consumption of the applications, as in low-power wireless neural recording tasks, signals must be compressed before transmission to extend battery life. In these cases, the CS can be used and it was demonstrated its potential in neural recording applications [95–97].

#### 2.3.2.3 Communication Systems

**Cognitive Radios** Cognitive radios (CRs) aim to provide a solution to the inefficient usage of the frequency spectrum. Spectrum sensing techniques suffer from computational complexity, hardware cost, and high processing time [57]. Since usually only some of the available channels are occupied by the users, the signal of interest is normally sparse in the frequency domain. Hence, the CS can be used to sense a wider spectrum with reduced sampling requirements, resulting in more power efficient systems [98–101].

**Sparse Channel Estimation** As said in Section 1.1, channels of several communication systems can be considered or well modelled as sparse channels. In these cases, better results can be achieved using the compressive sensing theory to estimate these channels [10, 18, 102, 103]. For instance, a low-complexity CS hardware implementation for channel estimation in the integrated services digital broadcasting-terrestrial (ISDB-T) system is proposed in FPGA in [104].

**Analog to Information Converter** The analog to digital converter (ADC) is based on the Nyquist sampling theorem in order to have a perfectly reconstruction of the information. That is, the signal is uniformly sampled at a rate at least twice its bandwidth. In several applications, the information of the signal is much smaller than its bandwidth. In these cases, this represents a waste of hardware and software resources to sample the whole signal. To deal with this, an analog to information converter (AIC) can use the CS theory to acquire a large bandwidth with relaxed sampling rate requirements, enabling faster, less expensive, and more energy-efficient solutions [105–109]. Examples of AIC are: random demodulator [108, 110, 111], modulated wideband converter [112] and non-uniform sampling [107, 113–115]. All these architectures have advantages and limitations. While the random demodulator AIC employs finite temporal sampling functions with infinite spectral support, the modulated wideband converter AIC has finite spectral sampling functions with infinite temporal support. Moreover, the modulated wideband converter AIC requires a large number of branches, so synchronization among the branches is also required, thus consuming more area and power. On the other hand, the non-uniform sampling AIC is sensitive to timing jitter, i.e., a sampling time with a small error can lead to a big error in the sample value for input signals that change rapidly.

#### 2.3.2.4 Detection and Recognition Systems

**Speech Recognition** Dictionary of example speech tokens can be used to sparsely represent speech signals [116]. Moreover, the speech signal can have sparse representation for a suitable selection of sparse basis functions, but for the noise, it will be difficult to derive a sparse representation. Therefore, it is possible to exploit this characteristic and through the CS theory achieve a better speech recognition performance [116–118].

**Seismology** The compressive sensing theory has an important use in data acquisition, that is, situations when it is intricate to obtain a lot of samples, for example in the case of seismic data [119]. The layers of the Earth can be estimated by measuring the reflections of a signal from different layers of the Earth. However, this requires a large data collection that is a time-consuming and expensive process. To deal with this, several works have proposed the CS for different seismic applications [119–121].

**Direction-of-Arrival** Direction-of-Arrival (DOA) estimation is the process of determining which direction a signal impinging on an array has arrived from [122, 123]. Since there are only a few non-zeros in the spatial spectrum of array signals, which represent their corresponding spatial locations, this sparsity can be applied to the DOA estimation [124]. Hence, the compressive sensing theory can be applied to the problem of DOA estimation by splitting the angular region into $N$ potential DOAs, where only $s \ll N$ of the DOAs have an impinging signal (alternatively $N - s$ of the angular directions have a zero-valued impinging signal present) [125, 126]. These DOAs are then estimated by finding the minimum number of DOAs with a non-zero valued impinging signal that still gives an acceptable estimate of the array output [122, 127].

# Chapter 3

# Sparse Recovery Algorithms

This chapter discusses several sparse recovery algorithms that have been proposed in the last years.

Sparse recovery algorithms have to recover a sparse signal from an undersampled set of measurements. They are usually classified into three main categories: convex relaxation, non-convex optimization techniques, and greedy algorithms [24]. Fig. 3.1 shows the algorithms that will be addressed in this chapter and their classification into these categories. In addition to the description of each of them, the following text provides an analysis and comparison of their performances.

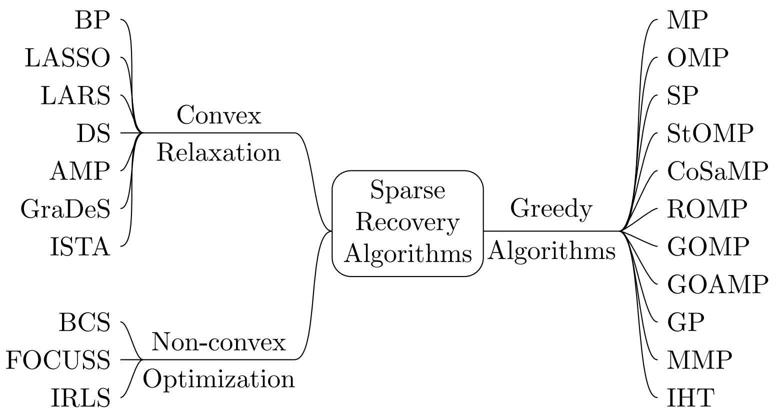


Figure 3.1 – Classification of sparse recovery algorithms.

Section 3.1 presents some algorithms from the first category (convex relaxation). These algorithms result in convex optimization problems whose efficient solutions exist relying on advanced techniques, such as projected gradient methods, interior-point methods, or iterative thresholding [18].

On the other hand, non-convex optimization approaches described in Section 3.2 can recovery the signal by taking into account a previous knowledge of its distribution [57]. Thanks to a posteriori probability density function, these solutions offer complete statistics of the estimate. Nonetheless, they can be unsuitable for high-dimensional problems due to their intensive computational requirements [128].

The third category is composed of the greedy algorithms. They recover the signal in an iterative way, making a local optimal selection at each iteration hoping to find the global optimum solution at the end of the algorithm (see Section 3.3).

## 3.1 Convex Relaxation

### 3.1.1 Basis Pursuit (BP)

Basis Pursuit (BP) is a signal processing technique that decomposes the signal into an superposition of basic elements. This decomposition is optimal in the sense that it leads to the smallest $l_1$ norm of coefficients among all such decompositions [29]. The BP algorithm seeks to determine a signal's representation that solves the problem:

$$\min ||\mathbf{h}||_1 \;\; subject \;\; to \;\; \mathbf{y} = \mathbf{Ah} \tag{3.1}$$

BP is a principle of global optimization without any specified algorithm. One of a possible algorithm to be used is the BP-simplex [29] that is inspired by the simplex method of linear programming [129]. For the BP-simplex, first, an initial basis $\mathbf{A}(\Lambda)$ is found by selecting $M$ linearly independent columns of $\mathbf{A}$. Then, at each step, the swap which best improves the objective functions is chosen to update the current basis, that is, one term in the basis is swapped for one term that is not in the basis [29].

In [130], the authors propose an algorithm for BP called Greedy Basis Pursuit (GBP). Unlike standard linear programming methods for BP, the GBP algorithm proceeds more like the MP algorithm, that is, it builds up the representation by iteratively selecting columns based on computational geometry [130]. Moreover, the GBP allows discarding columns that have already been selected [130].

### 3.1.2 BP De-Noising (BPDN) / Least Absolute Shrinkage and Selection Operator (LASSO)

The Basis Pursuit Denoising (BPDN) [29] / Least Absolute Shrinkage and Selection Operator (LASSO) [131] algorithm considers the presence of the noise **n**:

$$\min ||\mathbf{h}||_1 \;\; subject \;\; to \;\; \mathbf{y} = \mathbf{A}\mathbf{h} + \mathbf{n} \tag{3.2}$$

and aims to solve the optimization problem defined by:

$$\min(\frac{1}{2}||\mathbf{y} - \mathbf{A}\mathbf{h}||_2^2 + \lambda_p ||\mathbf{h}||_1) \tag{3.3}$$

where $\lambda_p > 0$ is a scalar parameter [29, 131]. The term $\lambda_p||\mathbf{h}||_1$ promotes sparseness of a reconstruction vector.

Its value greatly influences on the performance of the LASSO algorithm and therefore should be chosen carefully. In [29], the authors suggest:

$$\lambda_p = \sigma\sqrt{2\log(p)} \tag{3.4}$$

where $\sigma > 0$ is the noise level and $p$ is the cardinality of the dictionary [29].

Comparing with the LS cost function, it is possible to see that (3.3) basically includes a $l_1$ norm penalty term. Hence, under certain conditions, the solution would achieve the minimal LS error [132]. Since $||\mathbf{h}||_1$ is not differentiable for any zero position of **h**, it is not possible to obtain an analytical solution for the global minimum of (3.3).

There are several iterative techniques to find the minimum of (3.3) [29, 131]. One of these is called "Shooting" [133] and starts by the solution:

$$\hat{\mathbf{h}} = (\mathbf{A}^H\mathbf{A} + \mathbf{I})^{-1}\mathbf{A}^H\mathbf{y} \tag{3.5}$$

where **I** is the identity matrix. Let $\mathbf{a}_j$ be the $j^{th}$ column of the matrix **A** and $B_j$ be defined by (3.7), each $j^{th}$ element of $\hat{\mathbf{h}}$ is updated by:

$$\hat{h}(j) = \begin{cases} \frac{\lambda - B_j}{\mathbf{a}_j^T\mathbf{a}_j}, & if \quad B_j > \lambda \\ \frac{-\lambda - B_j}{\mathbf{a}_j^T\mathbf{a}_j}, & if \quad B_j < -\lambda \\ 0, & if \quad |B_j| \leq \lambda \end{cases} \tag{3.6}$$

$$B_j = -\mathbf{a}_j^T\mathbf{y} + \sum_{l\neq j}\mathbf{a}_j^T\mathbf{a}_l\hat{h}(l) \tag{3.7}$$

The original Shooting method is applied to real variables. For complex variables, an adaptation is necessary. In [134], two schemes are presented to adapt the LASSO algorithm to estimate a complex signal $\mathbf{h}$:

- r-LASSO: Let $imag(.)$ and $real(.)$ be the imaginary and real parts of a complex vector, respectively. It is defined by [134]:

$$\mathbf{y}^R = \begin{pmatrix} real(\mathbf{y}) \\ imag(\mathbf{y}) \end{pmatrix}, \quad \mathbf{h}^R = \begin{pmatrix} real(\mathbf{h}) \\ imag(\mathbf{h}) \end{pmatrix}, \quad \mathbf{A}^R = \begin{pmatrix} real(\mathbf{A}) & -imag(\mathbf{A}) \\ imag(\mathbf{A}) & real(\mathbf{A}) \end{pmatrix} \tag{3.8}$$

  These definitions are used in the Shooting method in (3.5) and each $j^{th}$ element of $\hat{\mathbf{h}}$ is calculated by [134]:

$$\hat{h}(j) = \hat{h}^R(j) + \sqrt{-1}\hat{h}^R(j+N) \tag{3.9}$$

- c-LASSO: The complex $l_1$-norm can be solved by some methods [134, 135]. It is defined by:

$$||\mathbf{h}||_1 = \sum_i |h(i)| = \sum_i \sqrt{real(h(i))^2 + imag(h(i))^2} \tag{3.10}$$

In many applications, the imaginary and real components tend to be either zero or non-zero simultaneously [134]. However, the r-LASSO does not take into account the information about any potential grouping of the real and imaginary parts [134]. On the other hand, the c-LASSO considers this extra information [134]. A comparison between r-LASSO and c-LASSO performed in [134] concludes that the c-LASSO outperforms the r-LASSO since it exploits the connection between the imaginary and the real parts.

### 3.1.3 Least Angle Regression (LARS)

The Least Angle Regression (LARS) algorithm begins with $\hat{\mathbf{h}} = \mathbf{0}$, the residual vector $\mathbf{b}_0 = \mathbf{y}$, and the active set $\Lambda = \emptyset$. This algorithm selects a new column from the matrix $\mathbf{A}$ at each iteration

$i$ and adds its index to the set $\Lambda_i$ [136]. The column $\mathbf{a}_{j_1}$ that has a smaller angle with $\mathbf{b}_0$ is selected at the first iteration. Then, the coefficient $\hat{h}_1(j_1)$ associated with the selected column $\mathbf{a}_{j_1}$ is increased [136]. Next, the smallest possible step in the direction of the column $\mathbf{a}_{j_1}$ is taken until another column $\mathbf{a}_{j_2}$ has as much absolute correlation value with the current residual as the column $\mathbf{a}_{j_1}$. The algorithm continues in a direction equiangular between the two active columns ($\mathbf{a}_{j_1}$,$\mathbf{a}_{j_2}$) until a third column $\mathbf{a}_{j_3}$ earns its way into the most correlated set [136]. The algorithm stops when no remaining column has correlation with the current residual [136].

Fig. 3.2 illustrates the begin of the LARS algorithm considering a two-dimensional system [137]. As said before, LARS starts with $\hat{\mathbf{h}}_0 = \mathbf{0}$ and the residual vector $\mathbf{b}_0 = \mathbf{y}$. Let $\theta_t(i)$ be the angle between the column $\mathbf{a}_{j_i}$ and the current residual vector $\mathbf{b}_i = \mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_i$ at iteration $i$, the column $\mathbf{a}_{j_1}$ is selected due to its absolute correlation with the initial residual vector compared to $\mathbf{a}_{j_2}$ ($\theta_1(1) < \theta_1(2)$) [137]. Next, the algorithm continues in the direction of $\mathbf{a}_{j_1}$ by adding the step size $\gamma_1$. $\gamma_1$ is chosen in a way to guarantee that the columns $\mathbf{a}_{j_1}$ and $\mathbf{a}_{j_2}$ have the same absolute correlation with the current residual vector at the next iteration ($\theta_2(1) = \theta_2(2)$). The solution coefficient is $\hat{h}_1(j_1) = \gamma_1$ [137]. The column $\mathbf{a}_{j_2}$ is added to the set $\Lambda$ at the second iteration, and the LARS continues in a equiangular direction with $\mathbf{a}_{j_1}$ and $\mathbf{a}_{j_2}$. Then, the step size $\gamma_2$ that leads to the vector $\mathbf{y}$ is added [137]. Finally, the solution coefficients are equal to: $\hat{h}_2(j_1) = \gamma_1 + \gamma_2 d_2(j_1)$ and $\hat{h}_2(j_2) = \gamma_2 d_2(j_2)$, where $\mathbf{d}_2$ is the updated direction at the second iteration that is equiangular with the active columns ($\mathbf{a}_{j_1}$, $\mathbf{a}_{j_2}$). The estimated vector $\hat{\mathbf{h}}$ is updated by multiplying the step size $\gamma$ with the updated direction $\mathbf{d}$ [137]. The algorithm continues until the residual be zero.

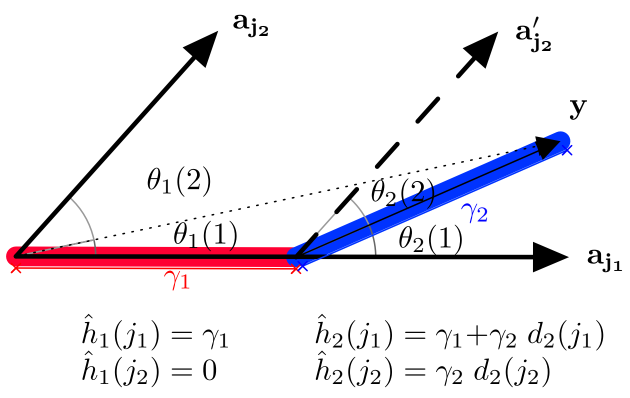


Figure 3.2 – LARS approximates the vector $\mathbf{y}$ by using $\mathbf{a}_{j_1}$ and $\mathbf{a}_{j_2}$.

A modified LARS called “homotopy algorithm” was proposed by Donoho and Tsaig to find a sparse solution of an underdetermined linear system [69].

These steps can summarize the LARS algorithm [137]:

- Step 1: Initialize the residual vector $\mathbf{b}_0 = \mathbf{y}$, the active set $\Lambda = \emptyset$, $\hat{\mathbf{h}}_0 = \mathbf{0}$ and the iteration counter $i = 1$.
- Step 2: Calculate the correlation vector: $\mathbf{c}_i = \mathbf{A}^T \mathbf{b}_{i-1}$.
- Step 3: Find the maximum absolute value in the correlation vector: $\lambda_i = ||\mathbf{c_i}||_\infty$.
- Step 4: Stop the algorithm if $\lambda \approx 0$. If not, go to Step 5.
- Step 5: Find the active set: $\Lambda = \{j : |c_i(j)| = \lambda_i\}$.
- Step 6: Solve the following least square problem to find active entries of the updated direction: $\mathbf{A}^T(\Lambda)\mathbf{A}(\Lambda)\mathbf{d}_i(\Lambda) = sign(\mathbf{c}_i(\Lambda))$.
- Step 7: Set the inactive entries of the updated direction to zero: $\mathbf{d}_i(\Lambda^C) = \mathbf{0}$.
- Step 8: Calculate the step size $\gamma_i$ by:

$$\gamma_i = \min_{j \in \Lambda^c} \left\{ \frac{\lambda_i - c_i(j)}{1 - \mathbf{a}_j^T \mathbf{A}(\Lambda)\mathbf{d}_i(\Lambda)}, \frac{\lambda_i + c_i(j)}{1 + \mathbf{a}_j^T \mathbf{A}(\Lambda)\mathbf{d}_i(\Lambda)} \right\}$$

- Step 9: Calculate $\hat{\mathbf{h}}_i = \hat{\mathbf{h}}_{i-1} + \gamma_i \mathbf{d}_i$.
- Step 10: Update $\mathbf{b}_i = \mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_i$.
- Step 11: Stop the algorithm if $||\mathbf{b}_i||_2 < \varepsilon$. Otherwise, set $i = i + 1$ and return to Step 2.

### 3.1.4 The Dantzig Selector (DS)

The Dantzig Selector (DS) is a solution to $l_1$ minimization problem [138]:

$$\min ||\hat{\mathbf{h}}||_1 \;\; subjet \;\; to \;\; ||\mathbf{A}^T\mathbf{b}||_\infty \leq \sqrt{1+\delta_1}\lambda_N \sigma \tag{3.11}$$

where $\mathbf{b} = \mathbf{y} - \mathbf{A}\hat{\mathbf{h}}$ is the residual vector, $\sigma$ is the standard deviation of the Additive White Gaussian Noise in (2.8), $\lambda_N > 0$ and all the columns of $\mathbf{A}$ have norm less than $\sqrt{1+\delta_1}$. $||\mathbf{A}^T\mathbf{b}||_\infty$ is defined by:

$$||\mathbf{A}^T\mathbf{b}||_\infty = \sup_{1 \leq i \leq N} |(\mathbf{A}^T\mathbf{b})_i| \tag{3.12}$$

For an orthogonal matrix $\mathbf{A}$, the Dantzig Selector is the $l_1$-minimizer subject to the constraint $||\mathbf{A}^T\mathbf{y} - \hat{\mathbf{h}}||_\infty \leq \lambda_N\sigma$, and the $i^{th}$ element of $\hat{\mathbf{h}}$ is calculated by:

$$\hat{h}(i) = \max(|(\mathbf{A}^T\mathbf{y})_i| - \lambda_N\sigma, 0)sgn((\mathbf{A}^T\mathbf{y})_i) \tag{3.13}$$

### 3.1.5 Iterative Soft Thresholding (ISTA)

In [139], the authors demonstrate that soft thresholding can be used to minimize equations of the form:

$$\frac{1}{2}\,||\mathbf{Ah} - \mathbf{y}||_2^2 + \lambda\,||\mathbf{h}||_1 \tag{3.14}$$

The solution is given by the limit of the sequence, where each iteration is defined by [139]:

$$\hat{\mathbf{h}}_i = \eta_{st}\left(\hat{\mathbf{h}}_{i-1} + \beta\mathbf{A}^T\mathbf{b}_{i-1};\lambda\right), \;\; \mathbf{b}_i = \mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_i \tag{3.15}$$

where $\hat{\mathbf{h}}_0 = \mathbf{0}$, $\beta$ is a stepsize, and $\eta_{st}(.;\lambda)$ is the soft thresholding function defined by (3.16) applied to each element of the vector.

$$\eta_{st}(x;\lambda) = \begin{cases} x-\lambda, & if \quad x > \lambda \\ 0, & if \quad |x| \leqslant \lambda \\ x+\lambda, & if \quad x < -\lambda \end{cases} \tag{3.16}$$

Although ISTA is guaranteed to converge [139], it converges slowly. Therefore, several modifications have been proposed to speed it up such as the “fast ISTA” (FISTA) [140] and the neural network LISTA [34] (see Section 4.2.1).

### 3.1.6 Approximate Message Passing (AMP)

The Approximate Message Passing (AMP) algorithm is described in [141, 142]. This algorithm starts by $\hat{\mathbf{h}}_0 = \mathbf{0}$ and $\mathbf{b}_0 = \mathbf{y}$. Then, in each iteration $i$, it updates these vectors by:

$$\hat{\mathbf{h}}_i = \eta_{i-1}(\hat{\mathbf{h}}_{i-1} + \mathbf{A}^T\mathbf{b}_{i-1}) \tag{3.17}$$

$$\mathbf{b}_i = \mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_i + \frac{1}{\delta}\mathbf{b}_{i-1}\left\langle\eta'_{i-1}(\mathbf{A}^T\mathbf{b}_{i-1} + \hat{\mathbf{h}}_{i-1})\right\rangle \tag{3.18}$$

where $\delta = M/N$, $\eta_i(.)$ is the thresholding function, $\langle \mathbf{x} \rangle = \sum_{i=1}^{N} x(i)/N$ for $\mathbf{x} = (x(1), \ldots, x(N))$ and $\eta_i'(s) = \frac{\partial}{\partial s}\eta_i(s)$. The term $\frac{1}{\delta}\mathbf{b}_{i-1}\left\langle \eta_i'(\mathbf{A}^T\mathbf{b}_{i-1} + \hat{\mathbf{h}}_{i-1})\right\rangle$ is from theory of belief propagation in graphical model [141].

The thresholding function $\eta_i(.)$ depends on iteration and problem setting. In [142], the authors consider the threshold control parameter $\lambda$ and $\eta_i(.) = \eta(.;\lambda\sigma_i)$ defined by:

$$\eta(x;\lambda\sigma_i) = \begin{cases} (x - \lambda\sigma_i), & if \;\; x \geq \lambda\sigma_i \\ (x + \lambda\sigma_i), & if \;\; x \leq -\lambda\sigma_i \\ 0, & otherwise \end{cases} \tag{3.19}$$

where $\sigma_i$ is the mean square error of the current estimate solution $\hat{\mathbf{h}}_i$ at iteration $i$.

AMP outperforms ISTA in convergence speed, but it does not work well unless the matrix $\mathbf{A}$ has independent and identically distributed (i.i.d.) Gaussian entries ($\mathcal{N}(0, M^{-1})$) [39]. In order to overcome this restriction, the Orthogonal Approximate Message Passing (OAMP) was proposed in [143].

### 3.1.7 Gradient Descent with Sparsification (GraDeS)

This algorithm was proposed in [144]. It considers a measurement matrix $\mathbf{A}$ which satisfies the RIP with an isometric constant $\delta_{2s} < 1/3$. This algorithm finds a sparse solution for the $l_1$ minimization problem in an iterative way.

First, the algorithm initializes the signal estimation $\hat{\mathbf{h}}_0 = \mathbf{0}$. Then, in each iteration $i$, it estimates the signal by:

$$\hat{\mathbf{h}}_i = H_s\left(\hat{\mathbf{h}}_{i-1} + \frac{1}{\gamma}\mathbf{A}^H(\mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_{i-1})\right) \tag{3.20}$$

where $\gamma > 1$ and the operator $H_s()$ sets all components to zero except the $s$ largest magnitude components.

## 3.2 Non-Convex Optimization Techniques

### 3.2.1 Bayesian Compressive Sensing (BCS)

Let $\sigma^2$ be the noise variance, the sparse Bayesian learning (SBL) assumes the Gaussian likelihood model [145]:

$$p(\mathbf{y}|\mathbf{h};\sigma^2) = (2\pi\sigma^2)^{-M/2} \exp\left(\frac{-1}{2\sigma^2}||\mathbf{y}-\mathbf{A}\mathbf{h}||^2\right) \tag{3.21}$$

In a Bayesian formulation, the formalization that **h** is sparse is made by placing a sparseness-promoting prior on **h** [146]. The Laplace density function is a widely used sparseness prior [147, 148]:

$$p(\mathbf{h}|\lambda) = \left(\frac{\lambda}{2}\right)^N exp\left(-\lambda\sum_{i=1}^{N}|h_i|\right) \tag{3.22}$$

and henceforth the subscript $s$ on **h** is dropped, recognizing that the interest is in a sparse solution for the weights [146]. Thus, the solution of (2.8) corresponds to a maximum a posteriori (MAP) estimate for using the prior in (3.22) [131, 147].

According to the Bayesian probability theory, we consider that a class of prior probability distributions $p(\theta)$ is conjugate to a class of likelihood functions $p(x|\theta)$ if the resulting posterior distributions $p(\theta|x)$ are in the same family as $p(\theta)$ [146]. Since the Laplace prior is not conjugate to the Gaussian likelihood, the relevance vector machine (RVM) is used.

Assuming the hyperparameters $\alpha$ and $\alpha_0$ are known, a multivariate Gaussian distribution with mean and covariance given by (3.23) and (3.24) can express the posterior for **h** [146].

$$\mu = \alpha_0\Sigma\mathbf{A}^T\mathbf{y} \tag{3.23}$$

$$\Sigma = (\alpha_0\mathbf{A}^T\mathbf{A}+\mathbf{D})^{-1} \tag{3.24}$$

where $\mathbf{D} = diag(\alpha_1, \alpha_2, ...\alpha_N)$. Therefore, the search for the hyperparameters $\alpha$ and $\alpha_0$ can be seen as a learning problem in the context of the RVM. A type-II maximum likelihood (ML) procedure can be used to estimate these hyperparameters from the data [149].

The logarithm of the marginal likelihood for $\alpha$ and $\alpha_0$, noted $L(\alpha, \alpha_0)$, is given by [146]:

$$\begin{aligned}\log\ p(y|\alpha,\alpha_0) &= \log\int p(y|h,\alpha_0)p(h|\alpha)d\mathbf{h}\\ &= -\frac{1}{2}\left[M\log 2\pi+\log|\mathbf{C}|+\mathbf{y}^T\mathbf{C}^{-1}\mathbf{y}\right]\end{aligned} \tag{3.25}$$

with $\mathbf{C} = \sigma^2\mathbf{I}+\mathbf{A}\mathbf{D}^{-1}\mathbf{A}^T$. The maximization of (3.25) can be obtained with a type-II ML approximation that uses the point estimates for $\alpha$ and $\alpha_0$. This can be achieved through the Expectation-Maximization (EM) algorithm [146, 149], to yield:

$$\alpha_i^{new} = \frac{\gamma_i}{\mu_i^2} \tag{3.26}$$

where $\mu_i$ is the $i^{th}$ posterior mean weight from (3.23) and $\gamma_i = 1 - \alpha_i \Sigma_{ii}$ with $\Sigma_{ii}$ the $i^{th}$ diagonal element of (3.24).

### 3.2.2 Focal Underdetermined System Solution (FOCUSS)

The Focal Underdetermined System Solution (FOCUSS) was proposed in [150] to solve (2.8). First, a low-resolution initial estimate of the real signal is made. Then, the iteration process refines the initial estimate to the final localized energy solution [150]. The FOCUSS iterations are based on a weighted minimum norm solution defined as the solution minimizing a weighted norm $||\mathbf{W}^{-1}\mathbf{h}||_2$. It is given by [150]:

$$\hat{\mathbf{h}} = \mathbf{W}(\mathbf{A}\mathbf{W})^{\dagger}\mathbf{y} \tag{3.27}$$

where the definition of a weighted minimum norm solution is to find $\mathbf{h} = \mathbf{W}\mathbf{q}$ where $\mathbf{q} : min||\mathbf{q}||_2$, subject to $\mathbf{A}\mathbf{W}\mathbf{q} = \mathbf{y}$. The cost objective simply becomes $\left\|\mathbf{W}^{\dagger}\mathbf{h}\right\| = \sum_{i=1,w(i)\neq 0}^{N} \left(\frac{h(i)}{w(i)}\right)^2$ when $\mathbf{W}$ is diagonal, where $w_i$ are the diagonal entries of $\mathbf{W}$ [150].

The basis of the basic FOCUSS algorithm lies the Affine Scaling Transformation (AST):

$$\mathbf{q} = \hat{\mathbf{H}}_{k-1}^{\dagger}\hat{\mathbf{h}} \tag{3.28}$$

where $\hat{\mathbf{H}}_{k-1}^{\dagger} = diag(\hat{\mathbf{h}}_{k-1})$ [150]. Let $\mathbf{W}_{p_k}$ be the *a posteriori* weight in each iteration, the AST is used in the basic FOCUSS algorithm to construct the weighted minimum norm constraint (3.29) by setting $\mathbf{W}_{p_k} = \hat{\mathbf{H}}_{k-1}$ [150].

$$||\mathbf{W}^T\mathbf{h}||_2^2 = ||\mathbf{q}||_2^2 = \sum_{i=1,w_i\neq 0}^{n} \left(\frac{h(i)}{w(i)}\right)^2 \tag{3.29}$$

Let $\hat{\mathbf{h}}_0 = \mathbf{0}$, the steps of the algorithm are:

$$\text{Step 1:} \quad \mathbf{W}_{p_k} = (diag(\hat{\mathbf{h}}_{k-1})) \tag{3.30}$$

$$\text{Step 2:} \quad \mathbf{q}_k = (\mathbf{A}\mathbf{W}_{p_k})^{\dagger}\mathbf{y} \tag{3.31}$$

$$\text{Step 3:} \quad \hat{\mathbf{h}}_k = \mathbf{W}_{p_k}\mathbf{q}_k \tag{3.32}$$

The algorithm continues until a minimal set of the columns of **A** that describe **y** is obtained [150].

By introducing two parameters, the authors extend the basic FOCUSS into a class of recursively constrained optimization algorithms in [150]. In the first extension, $\hat{\mathbf{h}}_{k-1}$ is raised to some power $l$ [150]. While in the second extension an additional weight matrix $\mathbf{W}_{a_k}$ which is independent of the *a posteriori* constraints is used [150]. The follow steps describe the algorithm:

$$\text{Step 1:} \quad \mathbf{W}_{p_k} = (diag(\hat{\mathbf{h}}^{l}_{k-1})), \;\; l \in \mathbb{N}^{+} \tag{3.33}$$

$$\text{Step 2:} \quad \mathbf{q}_k = (\mathbf{A}\mathbf{W}_{a_k}\mathbf{W}_{p_k})^{\dagger}\mathbf{y} \tag{3.34}$$

$$\text{Step 3:} \quad \hat{\mathbf{h}}_k = \mathbf{W}_{a_k}\mathbf{W}_{p_k}\mathbf{q}_k \tag{3.35}$$

It can be assumed that $\mathbf{W}_{a_k}$ is constant for all iterations. According to [150], $l > 0.5$ when $h(i) > 0$ is imposed.

### 3.2.3 Iterative Reweighted Least Squares (IRLS)

The Iterative Reweighted Least Squares (IRLS) algorithm is used for solving (3.36) through a weigthed $l_2$ norm given by (3.37), where the weights are computated from the previous iterate $\mathbf{h}_{n-1}$, so $w(i) = |h_{n-1}(i)|^{p-2}$ [151].

$$\min_{\mathbf{h}} ||h||_p^p \;\; subject\;\; to \;\; \mathbf{Ah} = \mathbf{y} \tag{3.36}$$

$$\min_{\mathbf{h}} \sum_{i=1}^{N} w(i)h^2(i) \;\; subject\;\; to \;\; \mathbf{Ah} = \mathbf{y} \tag{3.37}$$

Let $\mathbf{Q}_n$ be the diagonal matrix with entries $1/w(i) = |h_{n-1}(i)|^{2-p}$, the solution of (3.37) can be given by:

$$\mathbf{h}_n = \mathbf{Q}_n\mathbf{A}^T(\mathbf{A}\mathbf{Q}_n\mathbf{A}^T)^{-1}\mathbf{y} \tag{3.38}$$

To deal with the case $0 \leq p \leq 1$, where $w(i)$ will be undefined for $h_{n-1}(i) = 0$, the authors in [151] regularize the optimization problem by incorporating a small $\varepsilon > 0$:

$$w(i) = ((h_{n-1}(i))^2 + \varepsilon)^{p/2-1} \tag{3.39}$$

## 3.3 Greedy Algorithms

Several greedy algorithms follow the steps showed in Fig. 3.3 [30]. There are some differences in the choice of the quantity of the column in each iteration, that is, the way to choose the indices $j$ to compose the set $J_i$. For instance, the MP and the OMP algorithms only choose one column in each iteration. In contrast, the StOMP algorithm chooses all columns whose the projection value is bigger than the threshold value $t_S$. The calculation of the residual vector $\mathbf{b}_i$ and the estimation of the non-zero values of $\hat{\mathbf{h}}$ in each iteration are other differences between the algorithms. For example, the SP algorithm estimates $\hat{\mathbf{h}}$ only at the end of the algorithm as it is explained in the subsections below.

Table 3.1 summarizes the inputs, the calculation of the residual vector $\mathbf{b}_i$ and the signal estimate components $\hat{\mathbf{h}}_i$ in each iteration [30]. In the next subsections, the greedy algorithms presented in Fig. 3.1 are explained.

Table 3.1 – Main parameters and calculations of Greedy Algorithms.

| Algorithm | Inputs | $j$ | $\mathbf{b}_i$ | $\hat{\mathbf{h}}_i$ |
|---|---|---|---|---|
| MP | $\mathbf{A}$, $\mathbf{y}$ | $\max_j \|\|c(j)\|\|$ | $\mathbf{b}_{i-1} - \frac{(\mathbf{a}_{l_i}{}^H\mathbf{b}_{i-1})\mathbf{a}_{l_i}}{\|\|\mathbf{a}_{l_i}\|\|_2^2}$ | $\frac{(\mathbf{a}_{l_i}{}^H\mathbf{b}_{i-1})}{\|\|\mathbf{a}_{l_i}\|\|_2^2}$ |
| OMP | $\mathbf{A}$, $\mathbf{y}$ | $\max_j \|\|c(j)\|\|$ | $\mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$ | $\mathbf{A}(\Lambda_i)^\dagger\mathbf{y}$ |
| SP | $\mathbf{A}$, $\mathbf{y}$, $s$ | $s$ biggest $\|\|c(j)\|\|$ | $\mathbf{y} - \mathbf{A}(\Lambda_i)\mathbf{A}^\dagger(\Lambda_i)\mathbf{y}$ | – |
| StOMP | $\mathbf{A}$, $\mathbf{y}$, $T$, $t_S$ | $j : \|\|c(j)\|\| > t_S$ | $\mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$ | $\mathbf{A}(\Lambda_i)^\dagger\mathbf{y}$ |
| CoSaMP | $\mathbf{A}$, $\mathbf{y}$, $s$ | $2s$ biggest $\|\|c(j)\|\|$ | $\mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$ | $supp_s(\mathbf{A}(\Lambda_i)^\dagger\mathbf{y})$ |
| ROMP | $\mathbf{A}$, $\mathbf{y}$, $s$ | $s$ biggest $\|\|c(j)\|\|$ | $\mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$ | $\mathbf{A}(\Lambda_i)^\dagger\mathbf{y}$ |
| GOMP | $\mathbf{A}$, $\mathbf{y}$, $Q$, $s$ | $Q$ biggest $\|\|c(j)\|\|$ | $\mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$ | $\mathbf{A}(\Lambda_i)^\dagger\mathbf{y}$ |
| GOAMP | $\mathbf{A}$, $\mathbf{y}$, $Q$ | $Q$ biggest $\|\|c(j)\|\|$ | $\mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$ | $\mathbf{A}(\Lambda_i)^\dagger\mathbf{y}$ |
| GP | $\mathbf{A}$, $\mathbf{y}$ | $\max_j \|\|c(j)\|\|$ | $\mathbf{b}_{i-1} - a_i\mathbf{A}(\Lambda_i)\mathbf{d}_i$ | $\hat{\mathbf{h}}_{i-1} + a_i\mathbf{d}_i$ |

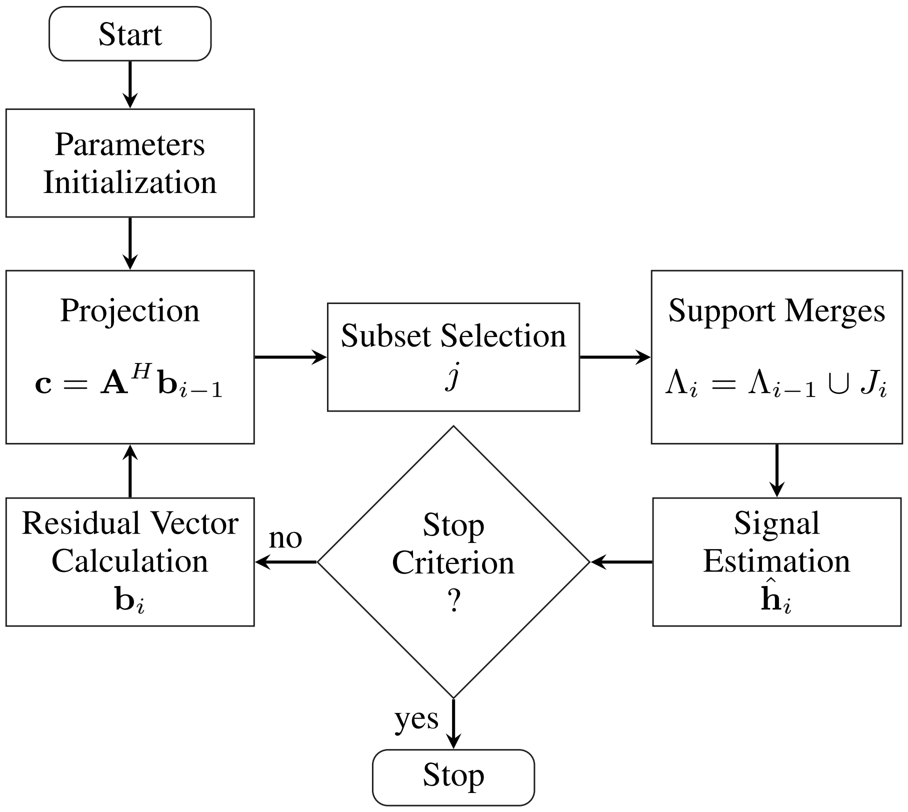

Start
Parameters Initialization
Projection
$\mathbf{c} = \mathbf{A}^H \mathbf{b}_{i-1}$
Subset Selection
$j$
Support Merges
$\Lambda_i = \Lambda_{i-1} \cup J_i$
Signal Estimation
$\hat{\mathbf{h}}_i$
Stop Criterion ?
no
Residual Vector Calculation
$\mathbf{b}_i$
yes
Stop

Figure 3.3 – Greedy Algorithms Diagram.

### 3.3.1 Matching Pursuit (MP)

The Matching Pursuit (MP) algorithm is proposed in [28]. Let $\hat{\mathbf{h}}_0 = \mathbf{0}$, each iteration $i$ of the MP algorithm consists in finding the column $\mathbf{a}_{k_i} \in \mathbf{A}$ which is best aligned with the residual vector $\mathbf{b}_{i-1}$ ($\mathbf{b}_0 = \mathbf{y}$) according to (3.40) [28].

$$k_i = \arg\max_{l} |\mathbf{a}_l{}^H \mathbf{b_{i-1}}|, \quad l = 1, 2, ...., N \tag{3.40}$$

The index set $\Lambda_i$ stores the indices of the best aligned columns after $i$ iterations. Let $\mathbf{D}_i$ be the matrix formed by the columns $\mathbf{a}_{k_i}$ chosen until iteration $i$, the next step is:

- $\Lambda_i = \Lambda_{i-1} \cup k_i$ and $\mathbf{D}_i = [\mathbf{D}_{i-1}, a_{k_i}]$, if $k_i \notin \Lambda_{i-1}$;
- $\Lambda_i = \Lambda_{i-1}$ and $\mathbf{D}_i = \mathbf{D}_{i-1}$, , if $k_i \in \Lambda_{i-1}$.

Then, a new residual vector is computed as (3.41) by removing the projection of $\mathbf{b}_{i-1}$ along this direction, and the estimated coefficient is calculated by (3.42).

$$\mathbf{b}_i = \mathbf{b}_{i-1} - P_{\mathbf{a}_{k_i}}\mathbf{b}_{i-1} = \mathbf{b}_{i-1} - \frac{(\mathbf{a}_{k_i}{}^H\mathbf{b}_{i-1})\mathbf{a}_{k_i}}{||\mathbf{a}_{k_i}||_2^2} \tag{3.41}$$

$$\hat{h}_i(k_i) = \hat{h}_{i-1}(k_i) + \frac{(\mathbf{a}_{k_i}{}^H\mathbf{b}_{i-1})}{||\mathbf{a}_{k_i}||_2^2} \tag{3.42}$$

The stop criterion of the algorithm can be, for example, $||\mathbf{b}_i|| \leq \varepsilon$. The signal estimate corresponds to the projections of the best columns of the matrix $\mathbf{A}$

### 3.3.2 Orthogonal Matching Pursuit (OMP)

The Orthogonal Matching Pursuit (OMP) algorithm is an improvement of the MP [152]. It can be stated as follows:

- Step 1: Initialize $\mathbf{b}_0 = \mathbf{y}$, $\Lambda_i = \emptyset$, and $i = 1$.
- Step 2: Find $l$ that solves the maximization problem $\max_l ||P_{a_l}\mathbf{b_{i-1}}||_2 = \max_l \frac{\mathbf{a}_l{}^H\mathbf{b}_{i-1}}{||\mathbf{a}_l||_2^2}$ and update $\Lambda_i = \Lambda_{i-1} \cup \{l\}$.
- Step 3: Calculate $\hat{\mathbf{h}}_i = \mathbf{A}^\dagger(\Lambda_i)\mathbf{y}$ and update $\mathbf{b}_i = \mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$.
- Step 4: Stop the algorithm if the stopping condition is achieved (e.g. $||\mathbf{b}_i|| \leq \varepsilon$). Otherwise, set $i = i + 1$ and return to Step 2.

In the OMP, the residual vector $\mathbf{b}_i$ is always orthogonal to the columns that have already been selected. Therefore, there will be no columns selected twice and the set of selected columns is increased through the iterations.

Due to the OMP selects only one column in each iteration, it is very sensitive to the selection of the index [153]. Alternatively, various approaches investigating multiple columns chosen in each iteration have been proposed such as the SP, the StOMP, the CoSaMP, the ROMP, the GOMP, the GOAMP, the MMP, and the GP algorithms. They are presented in the next sections.

### 3.3.3 Subspace Pursuit (SP)

At each stage, in order to refine an initially chosen estimate for the subspace, the Subspace Pursuit (SP) algorithm tests subsets of $s$ columns in a group [154]. That is, maintaining $s$ columns of $\mathbf{A}$, the algorithm executes a simple test in the spanned list of space, and after refines the list by discarding the unreliable candidates, retaining reliable ones while adding the same number of new candidates [154]. Basically, the steps of the SP are:

- Step 1: Initialize the support set $\Lambda_0$ with the $s$ indices corresponding to the largest magnitude entries in the vector $\mathbf{A}^H\mathbf{y}$, the residual vector $\mathbf{b}_0 = \mathbf{y} - \mathbf{A}(\Lambda_0)\mathbf{A}(\Lambda_0)^\dagger\mathbf{y}$ and the iteration counter $i = 1$.
- Step 2: $\hat{\Lambda}_i = \Lambda_{i-1} \cup J_i$, where $J_i$ is the set of the $s$ indices corresponding to the largest magnitude entries in the vector $\mathbf{c}_i = \mathbf{A}^H\mathbf{b}_{i-1}$.
- Step 3: Calculate $\mathbf{x}_i = \mathbf{A}^\dagger(\hat{\Lambda}_i)\mathbf{y}$.
- Step 4: Update $\Lambda_i = \{$s indices corresponding to the largest magnitude elements of $\mathbf{x}_i\}$.
- Step 5: Update $\mathbf{b}_i = \mathbf{y} - \mathbf{A}(\Lambda_i)\mathbf{A}^\dagger(\Lambda_i)\mathbf{y}$.
- Step 6: Stop the algorithm if the stopping condition is achieved. Otherwise, set $i = i+1$ and return to Step 2.

After $T$ iterations, the signal estimated is given by $\hat{\mathbf{h}} = \mathbf{A}^\dagger(\Lambda_T)\mathbf{y}$.

When the signal is very sparse, the SP algorithm has computational complexity upper-bounded by $O(sMN)$ ($s \leq const.\sqrt{N}$), that is, lower computational complexity than the OMP algorithm [154]. However, when the non-zero components of the sparse signal decay slowly, the computational complexity of the SP can be further reduced to $O(MNlogs)$ [154].

### 3.3.4 Stagewise Orthogonal Matching Pursuit (StOMP)

The Stagewise Orthogonal Matching Pursuit (StOMP) [155] algorithm is inspired by the OMP. Different from the OMP algorithm, the StOMP algorithm selects multiple columns at each iteration. That is, according to a threshold, the StOMP algorithm selects the subspaces composed of the columns with the highest coherence between the remaining columns and the residual vector [155]. The number of iterations is fixed.

The input parameters are: the number of iterations $T$ to perform, the threshold value $t_S$, the received signal $\mathbf{y}$, and the measurement matrix $\mathbf{A}$. The StOMP algorithm can be stated as follows:

- Step 1: Initialize the residual vector $\mathbf{b}_0 = \mathbf{y}$, $\Lambda_0 = \emptyset$, and $i = 1$.

- Step 2: Find $\mathbf{a}_l$ that $||P_{a_l}\mathbf{b_{i-1}}|| > t_S$, that is, $\max_l \frac{\mathbf{a}_l{}^H \mathbf{b}_{i-1}}{||\mathbf{a}_l||^2} > t_S$ and add the $\mathbf{a}_l$ columns to the set of selected columns. Update $\Lambda_i = \Lambda_{i-1} \cup \{l\}$.

- Step 3: Let $\hat{\mathbf{h}}_i = \mathbf{A}(\Lambda_i)^\dagger \mathbf{y}$. Update $\mathbf{b}_i = \mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$.

- Step 4: If the stopping condition is achieved ($i = N_{it} = T$), stop the algorithm. Otherwise, set $i = i+1$ and return to Step 2.

### 3.3.5 Compressive Sampling Matching Pursuit (CoSaMP)

The Compressive Sampling Matching Pursuit (CoSaMP) algorithm is presented in [156] to mitigate the unstability of the OMP algorithm. Similarly to the OMP, it starts by initializing a residual vector as $\mathbf{b}_0 = \mathbf{y}$, the support set as $\Lambda_0 = \emptyset$, the iteration counter as $i = 1$, and additionally sets $\hat{\mathbf{h}}_0 = \mathbf{0}$. The CoSaMP performs these steps [156]:

- Step 1 - Identification: a proxy of the residual vector from the current samples is formed and the largest components of the proxy $\mathbf{c}_i = |\mathbf{A}^H \mathbf{b}_{i-1}|$ are located. The first $2s$ entries of $\mathbf{c}_i$ with largest absolute values are selected, and the indices selected compose $J_i$.

- Step 2 - Support merger: the set of newly identified components is united with the set of components that appears in the current approximation. $\Lambda_i = J_i \cup supp(\hat{\mathbf{h}}_{i-1})$ is defined as the augmentation of the support of the previous estimate $\hat{\mathbf{h}}_{i-1}$ with the $2s$ indices corresponding to the entries of $\mathbf{c}_i$ with largest absolute values.

- Step 3 - Estimation: a least squares problem to approximate the target signal on the merged set of components is solved. $\hat{\mathbf{x}}_i = \mathbf{A}(\Lambda_i)^\dagger \mathbf{y}$.

- Step 4 - Pruning: a new approximation by retaining only the largest entries in this least squares signal approximation is produced. $\hat{\mathbf{h}}_i$ is composed of the first $s$ entries of $\hat{\mathbf{x}}_i$ with largest absolute values.

- Step 5 - Sample update: update $\mathbf{b}_i = \mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$.

### 3.3.6 Regularized OMP (ROMP)

The Regularized OMP (ROMP) algorithm was proposed in [157]. Firstly, the ROMP algorithm initializes $\Lambda_0 = \emptyset$ and the residual vector $\mathbf{b}_0 = \mathbf{y}$. Then, during each iteration $i$, the ROMP performs these three steps:

- Step 1 - Identification: $\hat{\Lambda}_i = \{$s biggest indices in magnitude of the projection vector $\mathbf{c}_i = \mathbf{A}^H \mathbf{b}_{i-1}\}$.
- Step 2 - Regularization: Among all subsets $J_i \subset \hat{\Lambda}_i$ with comparable coordinates $|\mathbf{c}(l)| \leq 2|\mathbf{c}(j)|$ for all $l, j \in J_i$, choose $J_i$ with the maximal energy $||\mathbf{c}(J_i)||_2$.
- Step 3 - Updating: Add the set $J_i$ to the index set: $\Lambda_i = \Lambda_i \cup J_i$. Calculate $\hat{\mathbf{h}}_i = \mathbf{A}(\Lambda_i)^\dagger \mathbf{y}$ and update the residual vector $\mathbf{b}_i = \mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$.

The regularization step can be done in linear time. The running time of the ROMP is comparable to that of the OMP in theory, but it is often better than the OMP in practice [157].

### 3.3.7 Generalized Orthogonal Matching Pursuit (GOMP)

The Generalized Orthogonal Matching Pursuit (GOMP) algorithm is a direct extension of the OMP algorithm [158]. The GOMP selects $Q \geq 1$ largest correlation columns of the matrix $\mathbf{A}$ with the residual vector $\mathbf{b}$. When $Q = 1$, the GOMP becomes the OMP. Moreover, $Q \leq s$ and $Q \leq \sqrt{M}$. The steps of the GOMP are:

- Step 1: Initialize the residual vector $\mathbf{b}_0 = \mathbf{y}$, $\Lambda_0 = \emptyset$ and $i = 1$.
- Step 2: Find the $Q$ biggest $\mathbf{a}_{l_1}, .., \mathbf{a}_{l_Q}$ columns that solves the maximization problem $\max_k ||P_{a_{l_k}} \mathbf{b_{i-1}}||_2 = \max_k \frac{\mathbf{a}_{l_k}{}^H \mathbf{b}_{i-1}}{||\mathbf{a}_{l_k}||_2^2}$ and add the $\mathbf{a}_{l_i}$ columns to the set of selected columns. Update $\Lambda_i = \Lambda_{i-1} \cup \{l_1, ..., l_Q\}$.
- Step 3: Calculate $\hat{\mathbf{h}}_i = \mathbf{A}(\Lambda_i)^\dagger \mathbf{y}$. Update $\mathbf{b}_i = \mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$.
- Step 4: Stop the algorithm if the stopping condition is achieved ($N_{it} = \min(s, M/Q)$ or $||\mathbf{b}_i||_2 \leq \varepsilon$). Otherwise, set $i = i + 1$ and return to Step 2.

The complexity of the GOMP algorithm is approximately $2N_{it}MN + (2Q^2 + Q)N_{it}^2 M$ [158].

### 3.3.8 Generalized Orthogonal Adaptive Matching Pursuit (GOAMP)

The Generalized Orthogonal Adaptive Matching Pursuit (GOAMP) algorithm considers that the signal's sparsity is not known, so it adapts the variable $Q$ of the GOMP algorithm during the iterations [159]. Basically, the GOAMP inserts a new step after the update of the residual vector:

- Step 1: Initialize the residual vector $\mathbf{b}_0 = \mathbf{y}$, $\Lambda_0 = \emptyset$ and $i = 1$.

- Step 2: Find the $Q$ biggest $\mathbf{a}_{l_1},..,\mathbf{a}_{l_Q}$ columns that solve the maximization problem $\max_k ||P_{a_{l_k}}\mathbf{b_{i-1}}||_2 = \max_k \frac{\mathbf{a}_{l_k}{}^H\mathbf{b}_{i-1}}{||\mathbf{a}_{l_k}||_2^2}$ and add the $\mathbf{a}_{l_i}$ columns to the set of selected columns. Update $\Lambda_i = \Lambda_{i-1} \cup \{l_1,...,l_Q\}$.

- Step 3: Calculate $\hat{\mathbf{h}}_i = \mathbf{A}(\Lambda_i)^{\dagger}\mathbf{y}$. Update $\mathbf{b}_i = \mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_i$.

- Step 4: If $||\mathbf{b}_{i-1} - \mathbf{b}_i||_2^2/||\mathbf{b}_{i-1}||_2^2 < \varepsilon_2$, $Q = f(Q)$. Otherwise, go to Step 5.

- Step 5: Stop the algorithm if the stopping condition is achieved ($||\mathbf{b}_i||_2 \le \varepsilon_1$). Otherwise, set $i = i+1$ and return to Step 2.

where $f(Q)$ is a function that increases the value of $Q$. According to [159], $\varepsilon_2$ is about $0.7-0.9$.

### 3.3.9 Gradient Pursuit (GP)

The Gradient Pursuit (GP) algorithms were proposed in [160] as variations of the MP algorithm. In the GP, at iteration $i$, the signal estimate $\hat{\mathbf{h}}_i$ is:

$$\hat{\mathbf{h}}_i = \hat{\mathbf{h}}_{i+1} + \gamma_i \mathbf{d}_i \tag{3.43}$$

where $\mathbf{d}_i$ is the update direction and $\gamma_i$ is the optimal step size defined by:

$$\gamma_i = \frac{<\mathbf{b}_{i-1}, \mathbf{A}(\Lambda_i)\mathbf{d}_i>}{||\mathbf{A}(\Lambda_i)\mathbf{d}_i||} \tag{3.44}$$

In the MP and the OMP algorithms, the update direction is taken to be in the direction of the best aligned column of the matrix $\mathbf{A}$. In the OMP, once added, the column will not be selected again as the process of orthogonalisation ensures that all future residuals will remain orthogonal to all currently selected columns. However, in the MP and the GP the orthogonality is not ensured. Hence, it is possible select again the same column.

Each iteration $i$ consists in finding the column $\mathbf{a}_{l_i} \in \mathbf{A}$ which is best aligned with the signal vector residual $\mathbf{b}_{i-1}$. The GP algorithms perform the follow steps:

- Step 1: Initialize $\mathbf{b}_0 = \mathbf{y}$, $\Lambda_0 = \emptyset$ and $i = 1$.

- Step 2: Find $l_i$ that solves the maximization problem $\max_{l_i} ||P_{a_{l_i}}\mathbf{b_{i-1}}||_2 = \max_{l_i} \frac{\mathbf{a}_{l_i}{}^H\mathbf{b}_{i-1}}{||\mathbf{a}_{l_i}||_2^2}$. Update $\Lambda_i = \Lambda_{i-1} \cup \{l_i\}$.

- Step 3: Update the direction $\mathbf{d}_i$. Calculate $\gamma_i = \frac{<\mathbf{b}_{i-1}, \mathbf{A}(\Lambda_i)\mathbf{d}_i>}{||\mathbf{A}(\Lambda_i)\mathbf{d}_i||}$ and $\hat{\mathbf{h}}_i = \hat{\mathbf{h}}_{i-1} + \gamma_i \mathbf{d}_i$. Update $\mathbf{b}_i = \mathbf{b}_{i-1} - \gamma_i \mathbf{A}(\Lambda_i)\mathbf{d}_i$.

- Step 4: Stop the algorithm, if the stopping condition is achieved. Otherwise, set $i = i+1$ and return to Step 2.

There are three different methods for calculating the update direction $\mathbf{d}_i$ [160, 161]:

- Gradient Pursuit: uses the direction that minimizes $||\mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_{i-1}||_2$, that is:

$$\mathbf{d}_i = \mathbf{A}^T(\Lambda_i)\left(\mathbf{y} - \mathbf{A}(\Lambda_i)\hat{\mathbf{h}}_{i-1}(\Lambda_i)\right) \tag{3.45}$$

- Conjugate Gradient Pursuit: it is a directional optimization algorithm that is guaranteed to solve quadratic optimization problems in as many steps as the dimension of the problem [162]. Let $\phi(\mathbf{h}) = \frac{1}{2}\mathbf{h}^T\mathbf{G}\mathbf{h} - \mathbf{f}^T\mathbf{h}$ be the cost function to be minimized, this method chooses $\mathbf{d}_i$ that is $\mathbf{G}$-conjugate to all the previous directions, that is:

$$\mathbf{d}_i\mathbf{G}\mathbf{d}_k = 0, \ \ \forall k < i \tag{3.46}$$

  In this case, $\mathbf{G} = \mathbf{A}^T(\Lambda_i)\mathbf{A}(\Lambda_i)$. Let $\mathbf{D}_i$ be the matrix whose columns are the update directions for the first $i$ iterations and let $\mathbf{g}_i$ be the gradient of the the cost function in iteration $i$, the new update direction $\mathbf{d}_i$ in iteration $i$ is given by [162]:

$$\mathbf{d}_i = \mathbf{g}_i + \mathbf{D}_{i-1}\mathbf{f} \tag{3.47}$$

  where $\mathbf{f} = -\left(\mathbf{D}_{i-1}^T\mathbf{G}\mathbf{D}_{i-1}\right)^{-1}\left(\mathbf{D}_{i-1}^T\mathbf{G}\mathbf{g}_{i-1}\right)$.

  The OMP uses a full conjugate gradient solver at every iteration. Instead, in this method, only a directional update step occurs for each new added element.

- Approximate Conjugate Gradient Pursuit: the new direction is conjugate to the previous direction, but this can be extended to a larger number of directions:

$$\mathbf{d}_i = \mathbf{g}_i + \mathbf{d}_{i-1}f \tag{3.48}$$

The **G**-conjugacy implies that:

$$\langle (\mathbf{G}\mathbf{d}_{i-1}), (\mathbf{g}_i + b\mathbf{d}_{i-1}) \rangle = 0 \tag{3.49}$$

$$f = -\frac{\langle (\mathbf{A}(\Lambda_i)\mathbf{d}_{i-1}), (\mathbf{A}(\Lambda_i)\mathbf{g}_i) \rangle}{\|\mathbf{A}(\Lambda_i)\mathbf{d}_{i-1}\|_2^2} \tag{3.50}$$

### 3.3.10 Multipath Matching Pursuit (MMP)

With the help of the greedy strategy, the Multipath Matching Pursuit (MMP) algorithm executes the tree search [153]. First, the MMP algorithm searches multiple promising columns of the matrix **A** candidates and then it chooses one minimizing the residual in the final moment. The MMP algorithm can not be represented by Fig. 3.3. Let $L$ be the number of child paths of each candidate, $f_i^k$ be the $k^{th}$ candidate in the $i^{th}$ iteration, $F_i = \{f_i^1, ..., f_i^u\}$ be the set of candidates in the $i^{th}$ iteration and $|F_i|$ be the number of elements of $F_i$, $\Omega^k$ is the set of all possible combinations of $k$ columns in **A**, for example, if $\Omega = \{1,2,3\}$ and $k = 2$, then $\Omega^k = \{\{1,2\},\{1,3\},\{2,3\}\}$ [153].

Fig. 3.4 shows a comparison from an hypothetical choice of columns in the first three iterations of the OMP and the MMP algorithms [153]. In this figure, the OMP selects the column with index 2 in the first iteration, then the index 1 in the next iteration and in the third iteration, it selects the index 4. On the other hand, the MMP algorithm selects the index 2 and 4 in the first iteration, after for each index selected, the algorithm will select others $L = 2$ index in each iteration. Then, in the second iteration, it selects the index 1 and 5 for the index 2 and for the index 4, but it is not necessary to select the same index as can be noted in the third iteration where the MMP selects the index 4 and 5 for the $\{2,1\}$ composing $f_3^1 = \{2,1,4\}$ and $f_3^2 = \{2,1,5\}$, and the index 2 and 3 for the $\{4,1\}$ composing $f_3^1 = \{2,1,4\}$ and $f_3^5 = \{4,1,3\}$. Moreover, it can be noticed that although the number of candidates increases as an iteration goes on (each candidate brings forth multiple children), the increase is actually moderate since many candidates are overlapping in the middle of search as the case of $f_3^1$, $f_3^2$ and $f_3^3$ in Fig. 3.4 [153].

The residual vector of the $k^{th}$ candidate in the $i^{th}$ iteration is $\mathbf{b}_i^k = \mathbf{y} - \mathbf{A}(f_i^k)\hat{\mathbf{h}}_i^k$, where $\mathbf{A}(f_i^k)$ is the matrix **A** using only the columns indexed by $f_i^k$. Given the measurement matrix **A**, the received signal **y**, the signal's sparsity $s$ and the parameter $L$, the MMP follows the steps bellow:

- Step 1: Initialize $\mathbf{b}_0 = \mathbf{y}$, $F_0 = \emptyset$ and $i = 1$.

- Step 2: Set $F_i = \emptyset$, $u = 0$ and $k = 1$.

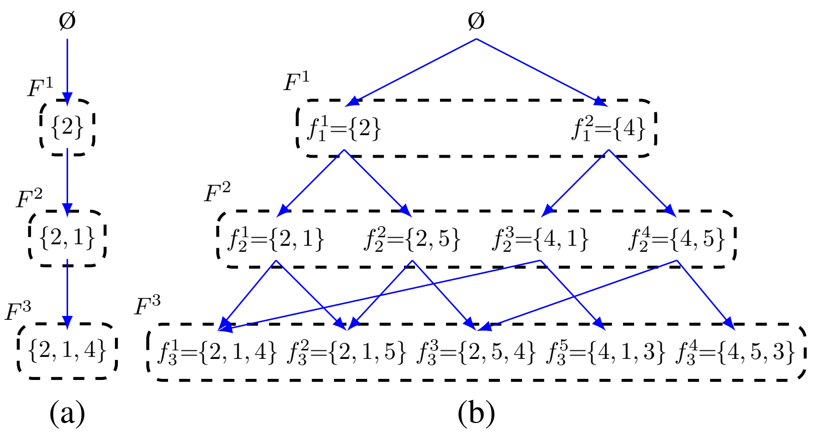


Figure 3.4 – Comparison between the OMP and the MMP algorithms ($L = 2$): (a) OMP (b) MMP.

- Step 3: Choose $L$ best indices of columns that solve the maximization problem $\arg\max ||\mathbf{A}^H\mathbf{b}_{i-1}^k||_2^2$ to compose $\pi$ and set $j = 1$.
- Step 4: Set $f^{temp} = f_{i-1}^k \cup \{\pi_j\}$, where $\pi_j$ is the $j^{th}$ element of the set $\pi$.
- Step 5: If $f^{temp} \notin F_i$ then $u = u+1$, $f_i^u = f^{temp}$, $F_i = F_i \cup \{f_i^u\}$, update $\hat{\mathbf{h}}_i^u = \mathbf{A}^\dagger(f_i^u)\mathbf{y}$ and $\mathbf{b}_i^u = \mathbf{y} - \mathbf{A}(f_i^u)\hat{\mathbf{h}}_i^u$. Otherwise, go to Step 6.
- Step 6: Set $j = j+1$. If $j \leq L$ then go to Step 4. Otherwise, go to Step 7.
- Step 7: Set $k = k+1$. If $k \leq |F_{i-1}|$ then go to Step 3. Otherwise, go to Step 8.
- Step 8: Set $i = i+1$. If $i > s$ then go to Step 9. Otherwise, go to Step 2.
- Step 9: Find the index of the best candidate, that is, $u^* = \arg\min_u ||\mathbf{b}_s^u||_2^2$. Set $\Lambda = f_s^{u^*}$ and calculate the estimate signal $\hat{\mathbf{h}} = \mathbf{A}^\dagger(\Lambda)\mathbf{y}$.

If the $\arg\max ||\mathbf{A}^H\mathbf{b}_{i-1}^k||_2^2$ in the Step 3 is calculated as in the OMP algorithm, the MMP algorithm is called Tree-based Orthogonal Matching Pursuit (TOMP) algorithm [163].

### 3.3.11 Iterative Hard Thresholding (IHT)

The Iterative Hard Thresholding (IHT) algorithm [164] is an iterative method that performs some thresholding function on each iteration. This algorithm can't be represented by Fig. 3.3. Let $\hat{\mathbf{h}}_0 = \mathbf{0}, i = 1$, for each iteration:

$$\hat{\mathbf{h}}_i = H(\hat{\mathbf{h}}_{i-1} + \mathbf{A}^H(\mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_{i-1}); \lambda) \tag{3.51}$$

where $H(x;\lambda)$ is defined by:

$$H(x;\lambda) = \begin{cases} 0, & if \quad |x| \leqslant \lambda \\ x, & if \quad |x| > \lambda \end{cases} \tag{3.52}$$

The IHT algorithm can stop after a fixed number of iterations or it can terminate when the sparse vector does not change much between consecutive iterations, for example [165].

In addition to the algorithms presented in the previous paragraphs, several others can be found in the literature, among them, Back-tracking based Adaptive Orthogonal Matching Pursuit (BAOMP) [166], Chaining Pursuit (CP) [167], Conjugate Gradient Iterative Hard Thresholding [168], Differential Orthogonal Matching Pursuit (D-OMP) [169], Denoising-based AMP (D-AMP) [170], Fast Iterative Shrinkage Thresholding Algorithm (FISTA) [140], Forward-Backward Pursuit (FBP) [171], Fourier sampling algorithm [172], Hard Thresholding Pursuit [173], Heavy Hitters on Steroids (HHS) [174], Iterative Half Thresholding Algorithm (IHTA) [175], Normalized Iterative Hard Thresholding [176], $l_p$-Regularized Least Squares Two Pass [177], Orthogonal Approximate Message Passing (OAMP) [143], Sequential Least Squares Matching Pursuit (SLSMP) [132], Sparse Adaptive Orthogonal Matching Pursuit (SpAdOMP) [178], Sparse Reconstruction by Separable Approximation (SpaRSA) [179], Stochastic Gradient Pursuit (SGP) [180], Stochastic Search Algorithms [181], Stein's Unbiased Risk Estimate Approximate Message Passing (SURE-AMP) [182], Tree Search Matching Pursuit (TSMP) [183], Vector Approximate Message Passing (VAMP) [184], Weighted Iterative Shrinkage Thresholding Algorithm (WISTA) [185]. Many of these algorithms are derivations of those presented in the previous sections. Therefore, it was preferred in this thesis to give more details to the ones that were used as the basis for the techniques proposed in Chapter 5 and as well as the basis for other algorithms proposed in the literature.

## 3.4 Performance Discussion

In order to have a better understanding of the potential of each solution, some algorithms representative of each category were selected for further study. The performance comparison between these algorithms resulted in a published scientific paper [30].

From the Convex Relaxation category, the AMP and FISTA algorithms were implemented. The BCS via RVM was implemented representing the Non-convex Optimization category. And finally, from the Greedy algorithms, the MP and OMP were implemented.

Let $N_s = 1000$ be the number of realizations, the normalized mean squared error (NMSE) described by (3.53) is used to evaluate the algorithms in terms of the size of the measured signal $y$ ($M$) and the signal's sparsity.

$$NMSE = \frac{1}{N_s}\sum \frac{\|\mathbf{h}-\hat{\mathbf{h}}\|_2^2}{\|\mathbf{h}\|_2^2} \tag{3.53}$$

The system model is defined by (2.8), considering:

- $N = 1024$;
- $SNR = 30\,\text{dB}$;
- $A$ is i.i.d. Gaussian, with elements distributed $\mathcal{N}(0, M^{-1})$;
- the sparse signal $\mathbf{h}$ to be estimated is Bernoulli-Gaussian, that is, its elements are i.i.d $\mathcal{N}(0,1)$ with probability $\gamma$ and the others are set to 0.

The results are compared to the theoretical performance bound Oracle Least Square (OLS), which has the previous knowledge of the non-zero tap positions. Let $\mathbf{A_s}$ be the matrix generated by $s$ columns of the matrix $\mathbf{A}$ related to the non-zero taps positions of the signal to be estimated, where $s$ is the sparsity of $\mathbf{h}$. The OLS calculates the non-zero coefficients of $\hat{\mathbf{h}}$ by applying the pseudo-inversion process in the matrix $\mathbf{A}_s$.

First, the algorithms performances are analyzed varying the size of $M$ for $\gamma = 0.05$ as shown in Fig. 3.5 [30]. It can be seen that the performances of all the algorithms increase when the number of measurements $M$ increases. However, it can be noticed that a low $M$ value ($M < N$) allows the algorithms to recover the sparse signal resulting in low NMSE values. Among the algorithms analyzed, the BCS presents the best performance. Furthermore, its performance is close to the one achieved by the OLS. It confirms the good results achieved by the algorithms from the Bayesian theory [30].

The algorithms performances are also analyzed varying $\gamma$ for $M = 512$ as shown in Fig. 3.6 [30]. According to Fig. 3.6, as the signal becomes less sparse (i.e. $\gamma$ increases), the performances of all algorithms decrease, that is, the NMSE values increase. When $\gamma$ is low, BCS is the algorithm that achieves the best performance (lower NMSE value), which is close to the one achieved by the OLS. However, when the signal to be estimated is less sparse (big $\gamma$ values), FISTA shows a better performance in recovering the signal [30].

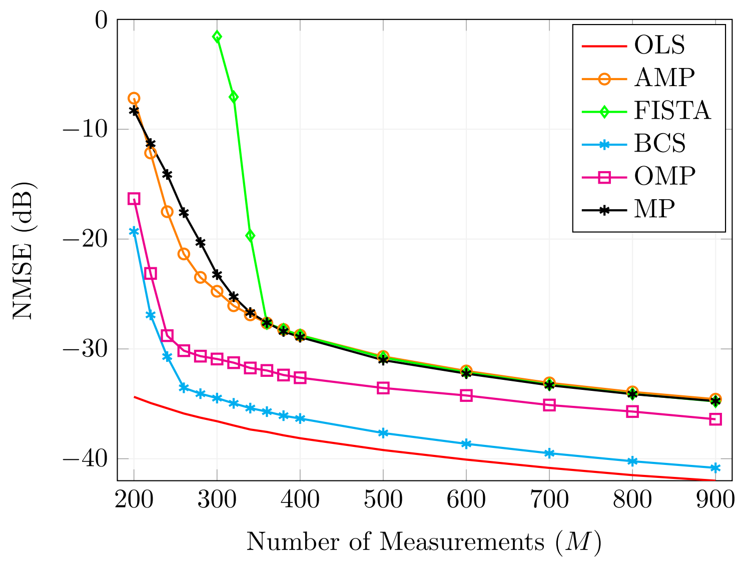


Figure 3.5 – Algorithms performances varying $M$ for $\gamma = 0.05$.

Table 3.2 shows the percentage of non-zero tap positions correctly found for the five algorithms analyzed [30]. The result of FISTA is not presented for $M = 200$ because in this scenario this algorithm did not converge. It can be observed that:

- when the $M$ value increases, the percentage of non-zero tap positions correctly found increases.

- although AMP and FISTA algorithms present the highest percentage values for $M = 400$ and $\gamma = 0.05$, the algorithms BCS and OMP are the ones that achieve the best results in terms of NMSE (see Fig. 3.5). It means that even if BCS and OMP correctly find less non-zero tap positions than the algorithms AMP and FISTA, BCS and OMP are better able to estimate the non-zero coefficients resulting in lower NMSE values.

- when the $\gamma$ value increases, that is, the signal become less sparse, the percentage of non-zero tap positions correctly found decreases. This occurs for all the algorithms analyzed and confirms what was suggested in Fig. 3.6.

Other performance comparisons between other algorithms can be found in the literature. Some of them are presented below.

A performance comparison of the SP, the OMP, the ROMP, the GOMP, and the GOAMP algorithms is made in [186] for the reconstruction of an image. The recovery performance was

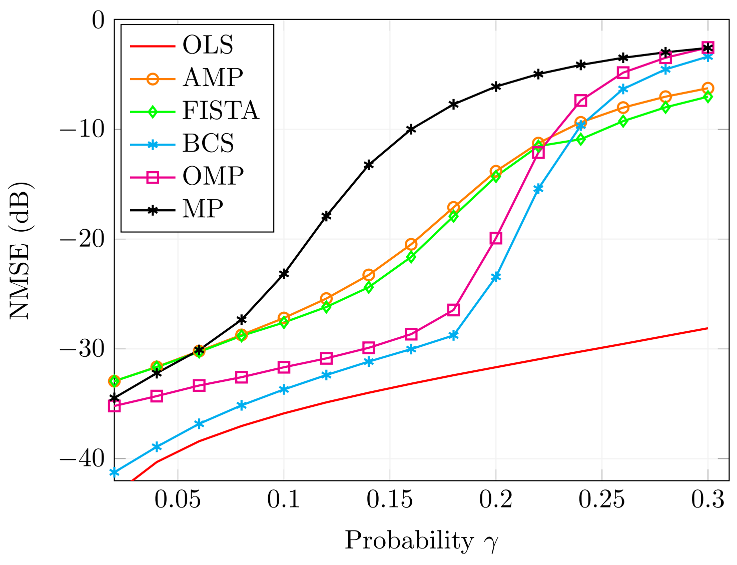


Figure 3.6 – Algorithms performances varying $\gamma$ for $M = 512$.

Table 3.2 – Percentage of non-zero tap positions correctly found.

| Algorithm | $\gamma = 0.05$ | | $\gamma = 0.1$ | $\gamma = 0.2$ |
|---|---|---|---|---|
| | $M = 200$ | $M = 400$ | $M = 512$ | |
| AMP | 81.7% | 98.4% | 97.7% | 88.5% |
| FISTA | - | 98.5% | 98.2% | 89.5% |
| BCS | 93.9% | 97.0% | 95.8% | 91.9% |
| OMP | 92.1% | 96.7% | 95.8% | 92.9% |
| MP | 67.6% | 96.8% | 96.2% | 70.1% |

analyzed in the form of Peak Signal to Noise Ratio (PSNR) value achieved and running time elapsed. From these simulations, the PSNR value is better when the GOAMP algorithm is used.

In [57], the authors compare the BCS, the BP, the GraDeS, the OMP, and the IHT algorithms to estimate a noisy sparse signal of length $N = 1024$. The metrics used were: phase transition diagram, recovery time, recovery error, and covariance. The results show that techniques of convex relaxation perform better in terms of recovery error, while greedy algorithms are faster, and Bayesian based techniques appear to have an advantageous balance of small recovery error and a short recovery time [57].

A comparison between the OMP and the modified LARS for solving LASSO algorithms is made in [137] considering the solution accuracy and the convergence time. The results show that generally the OMP requires fewer iterations than the LARS to converge to the final solution, suggesting that the OMP is much faster than the LARS [137]. However, for the cases where some columns of $\mathbf{A}$ are highly correlated, the OMP was considered less accurate than the LARS [137].

In [158], the authors compare the GOMP, the OMP, the StOMP, the ROMP, and the CoSaMP algorithms for a measurement matrix $\mathbf{A}$ $128 \times 256$ generated by a Gaussian distribution $N(0, 1/128)$. The sparse signal varies from $s = 1$ to $s = 70$ and it is generated in two ways: Gaussian signals and pulse amplitude modulation (PAM) signals. The results show that the critical sparsity of the GOMP algorithm is larger than that of the OMP, the ROMP, the StOMP, and the CoSaMP algorithms [158].

Algorithms OMP, StOMP, CoSaMP, MMP, and BPDN are compared in [153] varying the SNR for two different sparsity values ($s = 20$ and $s = 30$). The $100 \times 256$ measurement matrix is generated by a Gaussian distribution. The results show that the MMP performs close to the OMP to $s = 20$, but for $s = 30$, the performance of the MMP is better [153]. Moreover, the running time of these algorithms is shown as a function of $s$. The MMP algorithm has the highest running time and the OMP and the StOMP algorithms have the lowest running time among algorithms under test [153].

In [151], the authors compare the performance of the IRLS algorithm using the regularization. The results show that for $p = 1$ the unregularized IRLS and regularized IRLS are almost identical but for $p = 0$ and $p = 1/2$, the regularized IRLS algorithm recovers the greatest range of signals.

The authors in [163] compare the performance of the TOMP, the BP, and the OMP algorithms. According to their results, TOMP needs less iteration than the OMP because the TOMP algorithm selects the whole tree at a time and not only one element. Moreover, the TOMP can achieve better results than the BP and the OMP in reconstruction quality [163].

In [187], the authors implement the algorithms OMP and AMP in FPGA. As the OMP processing time increases quadratically with the number of non-zero coefficients of the signal to be estimated, this algorithm is more suitable to recover very sparse signals. On the other hand, if the signal to be estimated has several non-zero components it is more efficient to use the AMP algorithm to recover the signal than the OMP.

The study of these sparse recovery algorithms led to the proposition of a greedy algorithm called Matching Pursuit based on Least Squares (MPLS) that is presented in Section 5.1. In addition, the good results of FISTA and the simplicity of ISTA stimulated the study of the neural network LISTA and others neural networks used to sparse signal recovery (see Chapter 4).

# Chapter 4

# Deep Learning based Compressive Sensing

This chapter explores some neural networks used to estimate sparse signals. Firstly, key concepts related to deep learning are introduced. Then, several neural network proposed in the literature are addressed. Some of them were inspired by sparse recovery algorithms presented in Chapter 3.

## 4.1 Basics on Deep Learning

A neural network (NN) is a connection of several neuron elements to a layered architecture (see Fig. 4.1). In each neuron element, several inputs are weighted summed with bias and it is fed into an activation function $\sigma(.)$ [188]. Some activation functions are presented in Table 4.1 [188, 189].

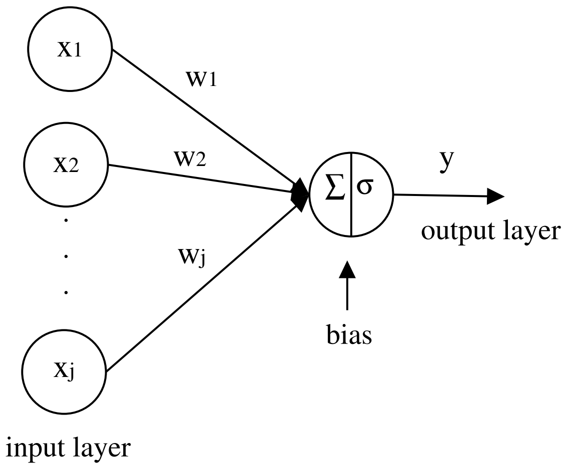


Figure 4.1 – A neuron element of a neural network.

Table 4.1 – Activation functions.

| Name | $\sigma(x)$ |
|---|---|
| ReLU | $max(0,x)$ |
| sigmoid | $\frac{1}{1+e^{-x}}$ |
| softmax | $\frac{e^{x_i}}{\sum_i e^{x_i}}$ |
| $tanh(x)$ | $\frac{e^x-e^{-x}}{e^x+e^{-x}}$ |

The feed-forward neural network is illustrated in Fig. 4.2. Its output in the layer $l$ is given by:

$$\mathbf{r}_l = \sigma(\mathbf{W}_l\mathbf{r}_{l-1} + \mathbf{e}_l) \tag{4.1}$$

where $\Theta_l = [\mathbf{W}_l, \mathbf{e}_l]$ is the set of learnable parameters of the layer $l$. $\sigma(.)$ introduces a non-linearity which is very important for the expressive power of the neural network [189].

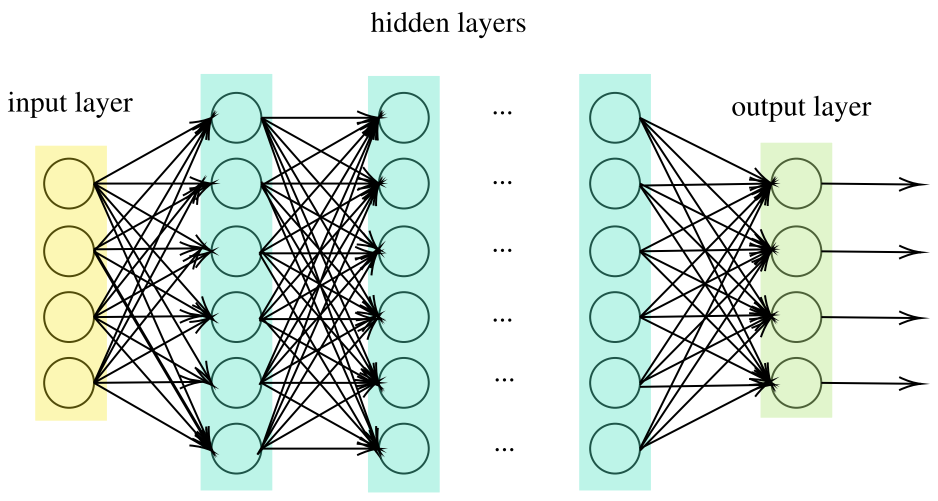


Figure 4.2 – A fully connected feed-forward NN architecture.

Recurrent Neural Network (RNN) is another NN (see Fig. 4.3). It can use its internal state (memory) to process sequences of inputs.

Another NN architechture is the Convolutional Neural Network (CNN). It adds convolutional and pooling layers before feeding into a fully connected network (see Fig. 4.4). In the convolutional layers, a dot production between a filter matrix and an input matrix comprised of several neurons generates each output. On the other hand, in the pooling layers, each output is obtained by the maximum value (max pooling) or the mean value (average pooling) of a group of neurons.

CNN is normally used for image recovery [190–195]. CNN works very effectively in recovering richly structured signals, however it does not appear to be well suited to sparse recovery

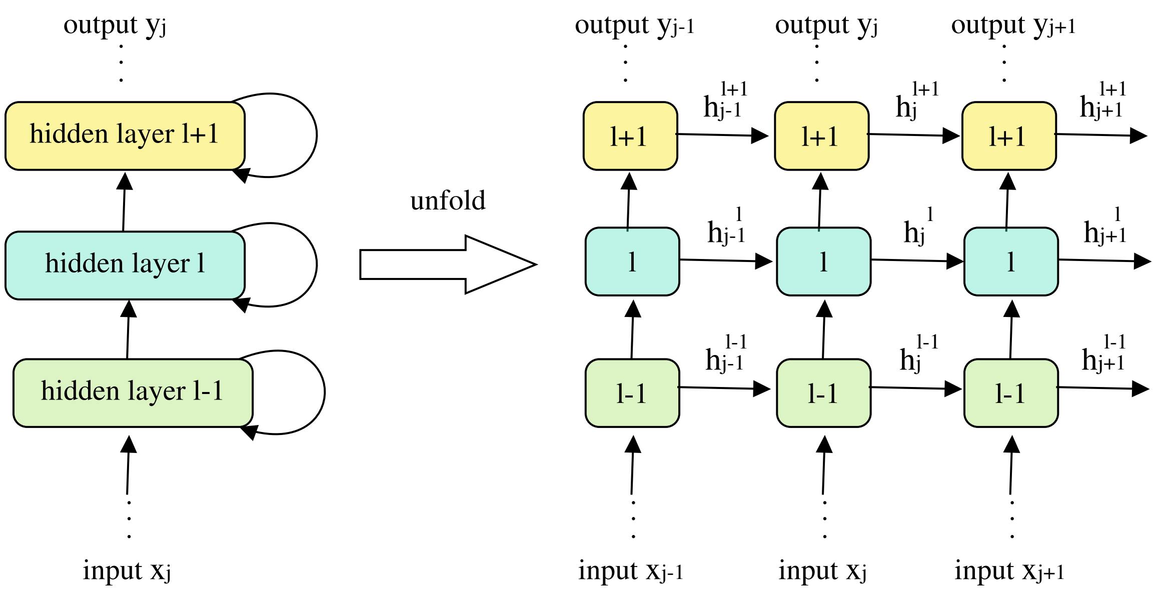


Figure 4.3 – A RNN architecture.

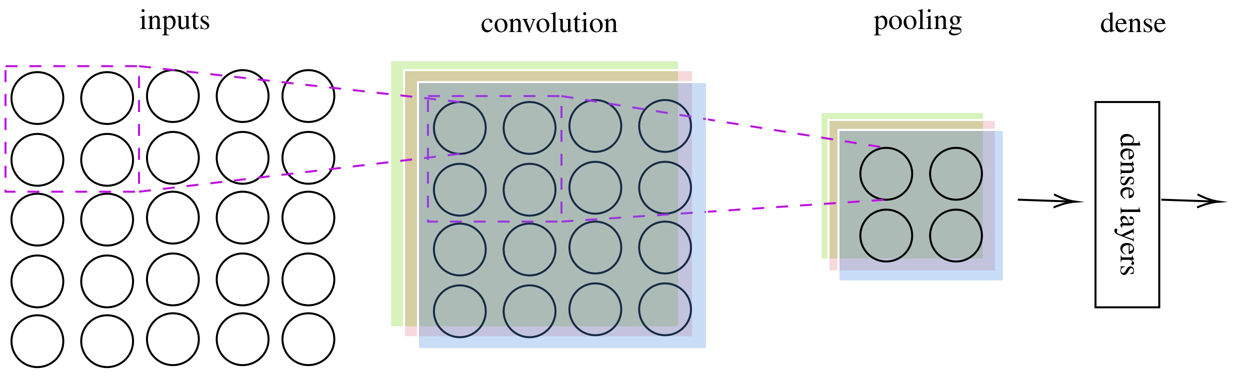


Figure 4.4 – A CNN architecture.

problems [31]. Indeed, in [31], the authors applied CNN to solve two 5G problems, but they did not obtain good results.

Once a neural network has been structured, it has to be trained. Given a set of training data, the NN is trained to minimize a loss function through an optimization algorithm, such as the Stochastic Gradient Descent (SGD) [196] and Adam [197]. There are two approaches to training: supervised and unsupervised.

Supervised training provides the network with the inputs and the corresponding outputs. In others words, the training data $\{(\mathbf{y}^d, \mathbf{h}^d)\}_{d=1}^{D}$ pairs based on signal statistics are used to train the parameters of the deep neural network (DNN) to accurately predict the unknown label $\hat{\mathbf{h}}$ associated with a newly observed feature $\mathbf{y}$. The NN compares its resulting outputs with the desired outputs and then it adjusts its parameters to minimize the error. After training the neural network with a set of $\{(\mathbf{y}^d, \mathbf{h}^d)\}_{d=1}^{D}$, the network can be used to predict the sparse signal $\mathbf{h}$ that corresponds to an input $\mathbf{y}$. The learned network is implicitly dependent on the matrix $\mathbf{A}$ and the noise statistics [31].

On the other hand, in an unsupervised training, the network has, in the training phase, the inputs without the corresponding outputs. For example Self-Organizing Maps which are also called Kohonen networks [198, 199]. Therefore, supervised learning requires prior training which makes difficult to implement the supervised NN in online processing and applications without training data. In contrast, unsupervised learning allows the use of NN for online processing when the learning procedure is fast enough.

The choice of the optimization algorithm, loss function, and activation function for a deep learning model influences on producing faster and accurate results. Moreover, the training data is very important to a good convergence of the NN.

Sparse recovery algorithms can be improved if they are combined with supervised deep learning. The literature reports different deep-learning approaches to solve sparse linear inverse problems [31–42, 185, 190, 191, 200–206]. For instance, it is possible to unroll/unfold a known iterative recovery algorithm into a neural network such as LISTA [34] (see Section 4.2.1) and LAMP [35] (see Section 4.2.2). These networks generally have numerical values as labels. Moreover, continuous and differentiable functions throughout the network structure allow the use of gradient-based learning methods to train network parameters [185]. Deep unfolding can be summarized as "given a model-based approach that requires an iterative inference method, we unfold the iterations into a layer-wise structure analogous to a neural network" [207]. A number of trainable parameters are learned using techniques from deep learning.

One of the main advantages of using neural network in compressive sensing is the reduction of computational complexity in the recovery step (CS reconstruction). Compared to sparse recovery algorithms, it means fewer matrix multiplications and iterations required to estimate the signal [35]. Instead of solving an optimization problem at the receiver, the network is defined before and its parameters are optimized to minimize a loss function during the training phase. Even if the training phase can require a lot of computational resources to better fit the network, in a lot of cases the neural network is only fit once up front and this phase is not necessary to run during the reception, so it can be done offline [203, 208].

## 4.2 Neural Networks Presented in the Literature

This section addresses some neural networks reported in the literature applied to sparse signals estimation. Some of them are based on the idea of unfolding an iterative algorithm. Fig. 4.5 shows the neural networks that will be discussed in more details in this section. The sparse recovery algorithms that inspired them are also illustrated in this figure.

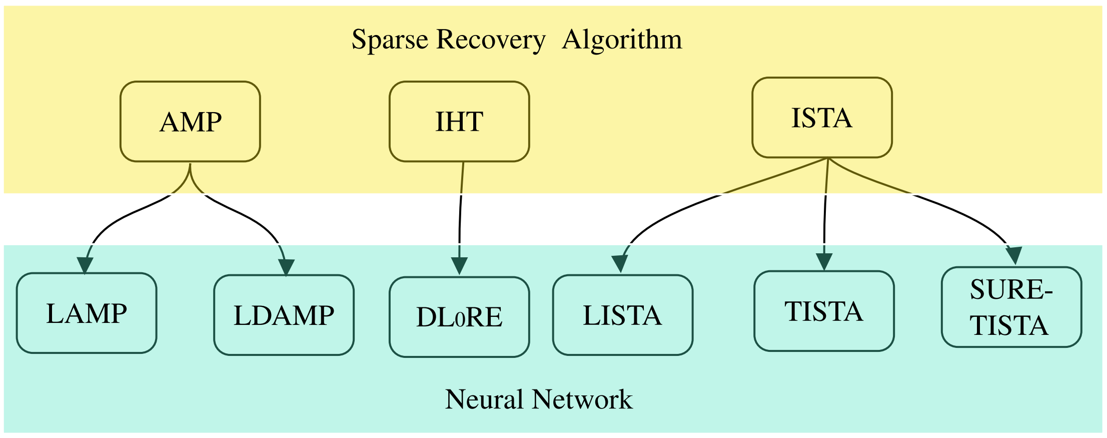


Figure 4.5 – Neural networks inspired by sparse recovery algorithms presented in this section.

### 4.2.1 Learned Iterative Shrinkage-Thresholding Algorithm (LISTA)

Inspired by the Iterative Shrinkage-Thresholding Algorithm (ISTA - see Section 3.1.5), a neural network architecture called Learned Iterative Shrinkage-Thresholding Algorithm (LISTA) was proposed in [34] to sparse signal estimation. Fig. 4.6 shows, as an example, the neural network LISTA with three layers.

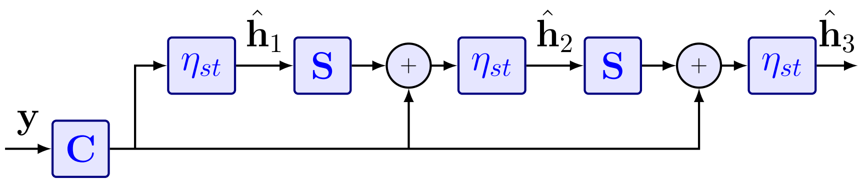


Figure 4.6 – Example of LISTA with 3 layers.

Let $\eta_{st}(x;\lambda)$ be defined by (3.16), the output $\hat{\mathbf{h}}_i$ of the $i-$layer of LISTA is calculated by:

$$\hat{\mathbf{h}}_i = \eta_{st}\left(\mathbf{S}\hat{\mathbf{h}}_{i-1} + \mathbf{C}\mathbf{y};\lambda_i\right) \tag{4.2}$$

LISTA "learns" the thresholds $\lambda_i$ and the matrices $\mathbf{C}$, $\mathbf{S}$ from the training data $\{(\mathbf{y}^d,\mathbf{h}^d)\}_{d=1}^{D}$

by minimizing the quadratic loss function [34]:

$$L_T(\Theta) = \frac{1}{D}\sum_{d=1}^{D} ||\hat{\mathbf{h}}_T(\mathbf{y}^d;\Theta) - \mathbf{h}^d||_2^2 \tag{4.3}$$

where $\lambda = [\lambda_1, \lambda_2, ..., \lambda_T]$ and $\Theta = [\mathbf{C}, \mathbf{S}, \lambda]$ denotes the set of learnable parameters and $\hat{\mathbf{h}}_T(\mathbf{y}^d;\Theta)$ the output of the $T-$layer network with input $\mathbf{y}^d$ and parameters $\Theta$.

After training the neural network with a set of $\{(\mathbf{y}^d, \mathbf{h}^d)\}_{d=1}^{D}$, the network can be used to predict the $\mathbf{h}$ that corresponds to the input $\mathbf{y}$.

To better understand the relation between ISTA and LISTA, we can rewrite (3.15) as (4.4):

$$\begin{aligned} \hat{\mathbf{h}}_i &= \eta_{st}\left(\hat{\mathbf{h}}_{i-1} + \beta\mathbf{A}^T\mathbf{b}_{i-1};\lambda\right) \\ &= \eta_{st}\left(\hat{\mathbf{h}}_{i-1} - \beta\mathbf{A}^T(\mathbf{A}\hat{\mathbf{h}}_{i-1} - \mathbf{y});\lambda\right) \\ &= \eta_{st}\left(\hat{\mathbf{h}}_{i-1} - \beta\mathbf{A}^T\mathbf{A}\hat{\mathbf{h}}_{i-1} + \beta\mathbf{A}^T\mathbf{y};\lambda\right) \end{aligned} \tag{4.4}$$

Then, considering $\beta\mathbf{A}^T = \mathbf{C}$ and $\mathbf{I}_N - \mathbf{C}\mathbf{A} = \mathbf{S}$, (4.4) can be rewritten as:

$$\begin{aligned} \hat{\mathbf{h}}_i &= \eta_{st}\left(\hat{\mathbf{h}}_{i-1} - \beta\mathbf{A}^T\mathbf{A}\hat{\mathbf{h}}_{i-1} + \beta\mathbf{A}^T\mathbf{y};\lambda\right) \\ &= \eta_{st}\left(\hat{\mathbf{h}}_{i-1} - \mathbf{C}\mathbf{A}\hat{\mathbf{h}}_{i-1} + \mathbf{C}\mathbf{y};\lambda\right) \\ &= \eta_{st}\left((\mathbf{I}_N - \mathbf{C}\mathbf{A})\hat{\mathbf{h}}_{i-1} + \mathbf{C}\mathbf{y};\lambda\right) \\ &= \eta_{st}\left(\mathbf{S}\hat{\mathbf{h}}_{i-1} + \mathbf{C}\mathbf{y};\lambda\right) \end{aligned} \tag{4.5}$$

From (4.4) and (4.5), it is possible to see the relation between ISTA (3.15) and LISTA (4.2). However, while in ISTA the matrix $\mathbf{A}$ is known, in LISTA instead of using the matrix $\mathbf{A}$, the matrices $\mathbf{C}$ and $\mathbf{S}$ are learned during the training phase.

LISTA as presented in (4.2) has to learn the parameters $\lambda_i$ for $i = 1, \ldots, N_L$, the N $\times$ M matrix $\mathbf{C}$, and the N $\times$ N matrix $\mathbf{S}$. Moreover, it does one matrix-vector multiplication with the matrix $\mathbf{S}$ in each layer. Therefore, the computational and memory complexity of LISTA are $\approx N_L N^2$ and $\approx N^2 + MN$, respectively.

LISTA can also be rewritten as (4.6) and illustrated by Fig. 4.7, where $\hat{\mathbf{h}}_0 = \mathbf{0}$ and $\hat{\mathbf{b}}_0 = \mathbf{y}$. The thresholds $\lambda_i$, the N $\times$ M matrix $\mathbf{C}$, and M $\times$ N matrix $\mathbf{G}$ are learned from the training data $\{(\mathbf{y}^d, \mathbf{h}^d)\}_{d=1}^{D}$ by minimizing the quadratic loss function (4.3). This configuration leads to memory complexity $\approx 2MN$. However, for each layer two matrix-vector multiplications are performed (with $\mathbf{C}$ and $\mathbf{G}$) leading to a computational complexity $\approx 2N_L MN$. If $\mathbf{G}$ is considered as $\mathbf{A}$, the computational and memory complexity decrease to $\approx N_L MN$ and $\approx MN$, respectively.

$$\hat{\mathbf{h}}_i = \eta_{st}\left(\hat{\mathbf{h}}_{i-1} + \mathbf{C}\mathbf{b}_{i-1}; \lambda_i\right), \quad \mathbf{b}_i = \mathbf{y} - \mathbf{G}\hat{\mathbf{h}}_i \tag{4.6}$$

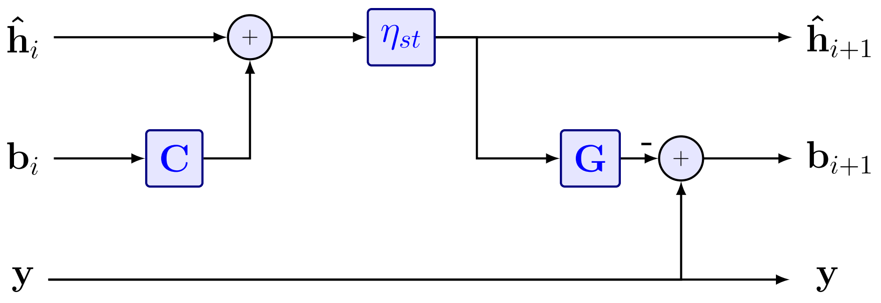


Figure 4.7 – One layer of LISTA network.

It can be noticed that for $2M < N$, LISTA represented by Fig. 4.7 and (4.6) results in lower computational and memory complexity than LISTA represented by Fig. 4.6 and (4.2). In addition, the computational complexity of one layer of LISTA is essentially equal to one iteration of ISTA or AMP algorithms [31]. Moreover, the estimates generated by the LISTA network need fewer matrix-vector multiplications than existing algorithms with optimally tuned regularization parameters ($\lambda$) [31]. LISTA can reach the results achieved by ISTA with much fewer layers than the number of ISTA iterations. This can be seen in Section 5.5.2 and in the results presented in [31], where LISTA took only 16 layers to reach an NMSE of $-35$ dB, whereas AMP took 25 iterations and ISTA took 4402 iterations.

In [200], the authors propose to use the cubic B-splines as shrinkage function in LISTA. Alternatively, the authors in [209] model the nonlinear activation function using a linear expansion of thresholds. In this thesis, other activation functions to be used in LISTA are analyzed in Section 5.5.

### 4.2.2 Learned Approximate Message Passing (LAMP)

The Learned Approximate Message Passing (LAMP) [31, 35] was inspired by AMP (see Section 3.1.6). Indeed, the neural network LAMP is constructed by unfolding the iterations of AMP.

Let $\lambda_i = \frac{\alpha_i ||\mathbf{b}_i||_2}{\sqrt{M}}$, the output $\hat{\mathbf{h}}_i$ of the $i-$layer of LAMP is calculated by (4.7). It can be noticed that the parameter $\lambda_i$ varies with the $\mathbf{b}_i$, while in LISTA it does not. LAMP also differs from LISTA in the presence of the last term in (4.8) known as "Onsager correction". The function $\eta$ depends on the problem setting as well as it occurs in AMP.

$$\hat{\mathbf{h}}_i = \eta(\hat{\mathbf{h}}_{i-1} + \mathbf{C}_{i-1}\mathbf{b}_{i-1}; \lambda_{i-1}) \quad (4.7)$$

$$\mathbf{b}_i = \mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_i + \frac{||\hat{\mathbf{h}}_i||_0}{M}\mathbf{b}_{i-1} \quad (4.8)$$

Fig. 4.8 illustrates one layer of LAMP. The parameters $\alpha_i$ and the N $\times$ M matrices $\mathbf{C}_i$ are learned during the training phase from the training data $\{(\mathbf{y}^d, \mathbf{h}^d)\}_{d=1}^{D}$. In each layer two matrix-vector multiplications are performed (with $\mathbf{C}_i$ and $\mathbf{A}$). Therefore, the computational and memory complexity of LAMP are $\approx 2N_LMN$ and $\approx N_LMN$, respectively. In [31], the authors also suggest one other version of LAMP where $\mathbf{C}$ is the same matrix in all layers resulting in a memory complexity $\approx MN$.

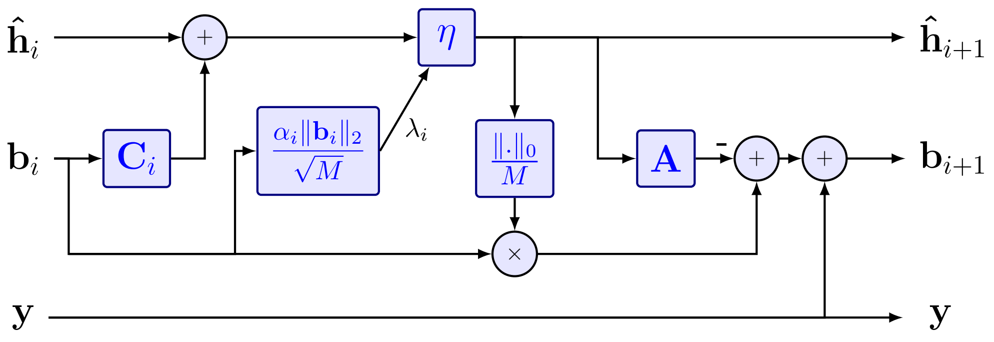


Figure 4.8 – One layer of the LAMP network.

### 4.2.3 Deep $l_0$-Regularized Encoder (D$l_0$RE)

Fig. 4.9 illustrates the Deep $l_0$-Regularized Encoder proposed in [36] where $\hat{\mathbf{h}}_i$ is the output of the layer $i$. The N $\times$ M matrix $\mathbf{C}$, the N $\times$ N matrix $\mathbf{S}$, and $\theta$ are shared among both stages and are learned during the training phase. In this thesis, this neural network is also called "D$l_0$RE".

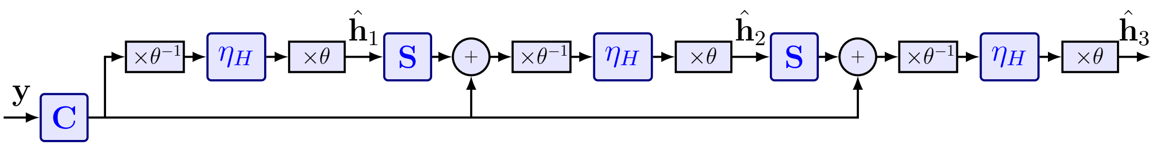


Figure 4.9 – Deep $l_0$-Regularized Encoder (D$l_0$RE).

The function $\eta_H$ defined by (4.9) is also called in [36] as Hard thrEsholding Linear Unit (HELU) function. It was chosen because it tends to produce highly sparse solutions and it does not penalize large values [36]. However, during the training phase, $\eta_H$ is replaced by HELU$_\sigma$

(when $\sigma \to 0$, HELU$_\sigma$ becomes $\eta_H$), that is, a continuous and piecewise linear function (see (4.10)). In [36], the authors start with $\sigma = 0.2$ and after each epoch, $\sigma$ is divided by 10.

$$\eta_H(x) = \begin{cases} 0, & if \quad |x| \leq 1 \\ x, & if \quad |x| \geq 1 \end{cases} \tag{4.9}$$

$$HELU_\sigma(x) = \begin{cases} 0, & if \quad |x| \leq 1-\sigma \\ \frac{(x-1+\sigma)}{\sigma}, & if \quad 1-\sigma < x < 1 \\ \frac{(x+1-\sigma)}{\sigma}, & if \quad -1 < x < \sigma - 1 \\ x, & if \quad |x| \geq 1 \end{cases} \tag{4.10}$$

D$l_0$RE performs one matrix-vector multiplication with $\mathbf{S}$ in each layer. Its computational and memory complexity are $\approx N_L N^2$ and $\approx N^2 + MN$, respectively. Comparing the IHT algorithm with D$l_0$RE, it can be seen some similarities. For instance, $\eta_H$ is the function (3.52) used in IHT with $\lambda = 1$. In addition, several similarities between LISTA (see Section 4.2.1) and the neural network illustrates in Fig. 4.9 can be noticed.

### 4.2.4 Learned Denoising-based Approximate Message Passing (LDAMP)

The Learned D-AMP (LDAMP) was proposed in [37] inspired by the D-AMP algorithm [170]. Let $\hat{\mathbf{h}}_i$ be the estimate of the signal $\mathbf{h}$ at iteration $i$ and $\hat{\mathbf{h}}_0 = \mathbf{0}$, each iteration $i$ of D-AMP calculates:

$$\mathbf{z}_i = \mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_i + \frac{\mathbf{z}_{i-1} divD_{\hat{\sigma}_{i-1}}(\hat{\mathbf{h}}_{i-1} + \mathbf{A}^H \mathbf{z}_{i-1})}{M} \tag{4.11}$$

$$\hat{\sigma}_i = \frac{||\mathbf{z}_i||_2}{\sqrt{M}} \tag{4.12}$$

$$\hat{\mathbf{h}}_{i+1} = D_{\hat{\sigma}_i}(\hat{\mathbf{h}}_i + \mathbf{A}^H \mathbf{z}_i) \tag{4.13}$$

where $\mathbf{z}_i$ is an estimate of the residual, $\hat{\sigma}_i$ is an estimate of the standard deviation of that noise, $D_{\hat{\sigma}_i}$ is the denoiser and $divD_{\hat{\sigma}_{i-1}}$ is the divergence of the denoiser [170].

Fig. 4.10 illustrates one layer of the LDAMP neural network [37]. The Denoiser block $D(.)$ is composed by a Denoising Convolutional Neural Network (DnCNN). The DnCNN neural network has 16 to 20 convolutional layers. The first layer uses 64 different $3x3xc$ filters (where $c$ is the number of color channels) and is followed by ReLU [37]. The next 14 to 18 convolutional layers each use 64 different $3x3x64$ filters which are each followed by batch-normalization [210] and a

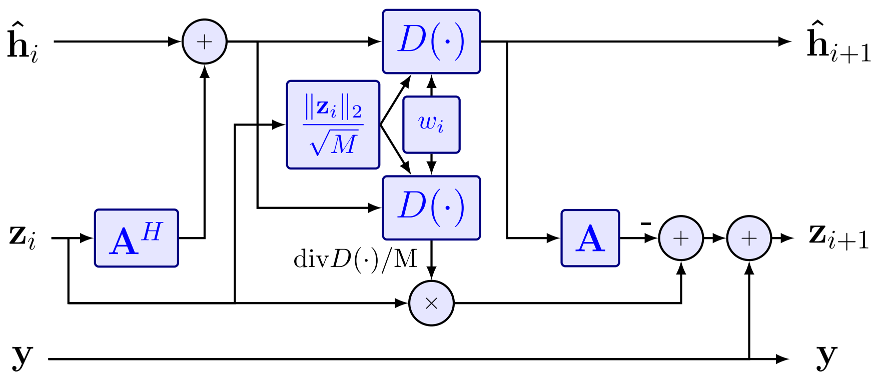


Figure 4.10 – One layer of the LDAMP network.

ReLU [37]. Finally, the last layer uses $c$ separate 3$x$3$x$64 filters to reconstruct the signal. Therefore, each layer $i$ of LDAMP can be described by:

$$\mathbf{z}_i = \mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_i + \frac{\mathbf{z}_{i-1} divD_{w_{i-1}(\hat{\sigma}_{i-1})}(\hat{\mathbf{h}}_{i-1} + \mathbf{A}^H \mathbf{z}_{i-1})}{M} \tag{4.14}$$

$$\hat{\sigma}_i = \frac{||\mathbf{z}_i||_2}{\sqrt{M}} \tag{4.15}$$

$$\hat{\mathbf{h}}_{i+1} = D_{w_i(\hat{\sigma}_i)}(\hat{\mathbf{h}}_i + \mathbf{A}^H \mathbf{z}_i) \tag{4.16}$$

In LDAMP, only the denoiser weights $w_i$ are learned, differently from LISTA (see Section 4.2.1), where the matrices $\mathbf{S}$ and $\mathbf{B}$ are learned during the training phase.

In [32], the authors consider the channel matrix as a 2D natural image and apply LDAMP which incorporates the denoising convolutional neural network (DnCNN) [192] for channel estimation in beamspace mmWave massive MIMO systems. The results indicate that the LDAMP network can achieve good performance for mmWave channel estimation.

### 4.2.5 Trainable ISTA (TISTA)

Trainable ISTA (TISTA) includes three parts: a linear estimator, a minimum mean squared error (MMSE) estimator-based shrinkage function, and a variance estimator [40, 41]. Fig. 4.11 illustrates one layer of TISTA.

The recursive formula of TISTA can be described by [40]:

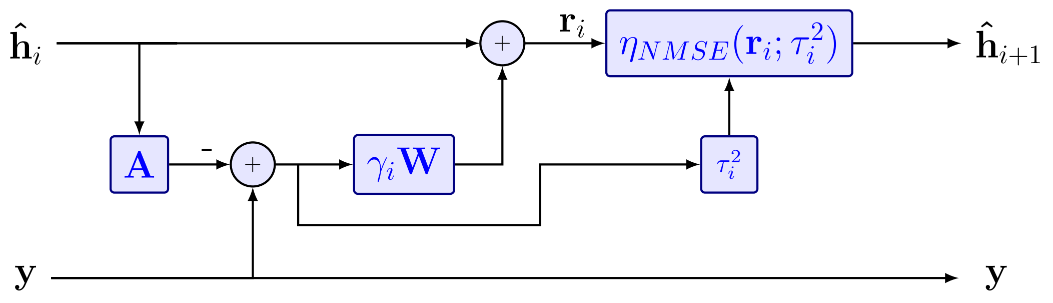


Figure 4.11 – One layer of the TISTA network.

$$\hat{\mathbf{h}}_{i+1} = \eta_{MMSE}(\mathbf{r}_i;\tau_i^2) \tag{4.17}$$

$$\mathbf{r}_i = \hat{\mathbf{h}}_i + \gamma_i\mathbf{W}(\mathbf{y}-\mathbf{A}\hat{\mathbf{h}}_i) \tag{4.18}$$

$$\tau_i^2 = \frac{v_i^2}{N}(N+(\gamma_i^2-2\gamma_i)M) + \frac{\gamma_i^2\sigma^2}{N}tr(\mathbf{W}\mathbf{W}^T) \tag{4.19}$$

$$v_i^2 = max\left\{\frac{||\mathbf{y}-\mathbf{A}\hat{\mathbf{h}}_i||_2^2 - M\sigma^2}{tr(\mathbf{A}^T\mathbf{A})}, \varepsilon\right\} \tag{4.20}$$

where $\hat{\mathbf{h}}_0 = \mathbf{0}$, $\sigma^2$ is the variance of the noise, $\mathbf{W} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$, $\gamma_i$ is a trainable parameter, and $\varepsilon$ is a small value (e.g. $\varepsilon = 10^{-9}$). The MMSE shrinkage function $\eta_{MMSE}$ is chosen according to the prior distribution of the original signal $\mathbf{h}$. The variables $\gamma_i$ control the variance of the MMSE shrinkage function.

The time complexity of TISTA per iteration is $O(N^2)$, which is the same time complexity of ISTA and AMP [40]. TISTA performs two matrix-vector multiplications (with $\mathbf{A}$ and $\mathbf{W}$) in each layer. Its computational and memory complexity are $\approx 2N_LMN$ and $\approx 2MN$, respectively.

Obtained results presented in [40] suggest that TISTA can be applied to different matrices $\mathbf{A}$ such as binary and Gaussian matrices. In addition, the resutls shows that TISTA converges faster than LISTA and AMP in several cases[40]. In [41], the authors propose two extensions of TISTA to deal with matrix $\mathbf{A}$ with nonzero-mean components or with a large condition number. The condition number $k$ of a matrix is defined as the ratio of the largest and smallest singular values, i.e., $k = s_1/s_M$ [41]. In addition, an extension of TISTA to complex-field nonlinear inverse problems is proposed in [42].

### 4.2.6 Stein's Unbiased Risk Estimate based-Trainable Iterative Thresholding Algorithm (SURE-TISTA)

Let a divergence-free denoiser $\hat{I}$ be constructed as (4.21) where $C$ is a constant and $I(.)$ is an arbitrary function [143].

$$\hat{I}(\mathbf{x}_t) = C\left(I(\mathbf{x}_t) - div\{I(\mathbf{x}_t)\mathbf{x}_t\}\right) \tag{4.21}$$

Stein's unbiased risk estimate based-trainable iterative thresholding algorithm (SURE-TISTA) was developed in [39] for sparse signal recovery problems. Fig. 4.12 illustrates a layer of SURE-TISTA. Each layer calculates:

$$\begin{aligned} \hat{\mathbf{h}}_i &= \hat{I}(\mathbf{r}_{i-1}; \lambda_i) \\ \mathbf{r}_{i-1} &= \hat{\mathbf{h}}_{i-1} + \mathbf{W}(\mathbf{y} - \mathbf{A}\hat{\mathbf{h}}_{i-1}) \end{aligned} \tag{4.22}$$

where $\hat{\mathbf{h}}_0 = \mathbf{0}$, $\mathbf{W} = \frac{N}{tr(\hat{\mathbf{W}}\mathbf{A})}\hat{\mathbf{W}}$, $\hat{\mathbf{W}} = \mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1}$, and $\lambda_i$ is a trainable parameter. $C_{optimal}$ is calculated based on SURE framework [39].

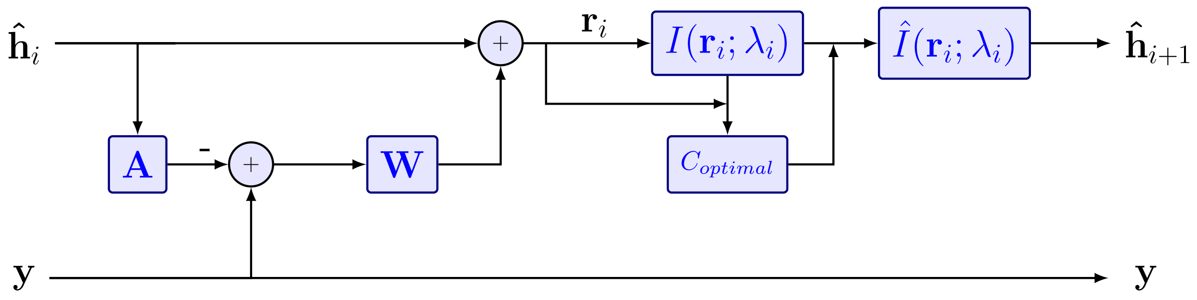


Figure 4.12 – One layer of the SURE-TISTA network.

As weel as TISTA, SURE-TISTA performs two matrix-vector multiplications (with $\mathbf{A}$ and $\mathbf{W}$) in each layer. Its computational and memory complexity are $\approx 2N_LMN$ and $\approx\ 2MN$, respectively.

The results presented in [39] indicate that SURE-TISTA requires fewer learnable variables to achieve similar performance as LAMP. Moreover, SURE-TISTA outperforms TISTA and does not require error measure estimators nor prior information [39].

In addition to neural networks discussed in the previous paragraphs, many others have been reported in the literature in the recent years. The list below enumerates other neural networks that can be used to sparse signal estimation: DeepInverse [201], Deep M-Sparse $l_0$ Encoder [36], Deep

Neural Network-structured IHTA (DNN-IHTA) [185], Deep Neural Network-structured WISTA (DNN-WISTA) [185], DR2-Net [190], DeepNIS [205], FompNet [206], Learned Vector Approximate Message Passing (LVAMP) [31], OAMP-Net [33], ReconNet [190], WDLReconNet [191]. These neural networks are not detailed here because some of them are more suitable for image estimation than for sparse channel estimation and others are variations from the NNs addressed in the sections above.

## 4.3 Discussion

This section presents a comparison between some NNs addressed in Section 4.2. Then an alternative solution to deal with complex numbers in NNs is exposed. Finally some challenges of the use of NN are discussed.

### 4.3.1 Comparison Between Some NNs

The choice of the NNs analyzed in this section was based on their representativeness or their use as a basis for other NNs. LISTA is one of the first neural network based on a sparse recovery algorithm presented in the literature. It inspired other authors to unfold known iterative recovery algorithms into neural network and apply it to sparse signal recovery.

These networks can outperform the results achieved by the original algorithm. For instance, one layer of LISTA, LAMP or TISTA has essentially the same computational complexity than one iteration of ISTA or AMP algorithms [31, 40]. However, sparse signal estimation using LISTA LAMP or TISTA can produce better estimates with fewer matrix-vector multiplications than using ISTA or AMP. Moreover, obtained results in [39] indicate that SURE-TISTA outperforms TISTA without require prior information nor error measure estimators.

Table 4.2 presents a comparison between some neural networks addressed in Section 4.2. In this table, $\Theta$ represents the set of parameters which are learned during the training phase, $M_c$ is the memory complexity, $MV$ is the number of matrix-vector multiplications performed in each layer, $C_c$ is the computational complexity, and $\sigma$ represents the activation function. The function $\eta$ in LAMP depends on the application (see Section 4.2.2). The LDAMP (see Section 4.2.4) is not presented in this table because its memory and computational complexity depend on the DnCNN chosen.

It can be notice that the quantity of learned parameters required in TISTA and SURE-TISTA is

Table 4.2 – Comparison of the neural networks addressed in Section 4.2.

| Neural Network | $\Theta$ | $M_c$ | $MV$ | $C_c$ | $\sigma$ |
|---|---|---|---|---|---|
| LISTA (Fig. 4.6) | $\mathbf{C}^{N\times M},\mathbf{S}^{N\times N},\lambda_1,\ldots,\lambda_{N_L}$ | $\approx N^2+MN$ | 1 | $\approx N_L N^2$ | $\eta_{st}$ (3.16) |
| LISTA (Fig. 4.7) | $\mathbf{C}^{N\times M},\mathbf{G}^{M\times N},\lambda_1,\ldots,\lambda_{N_L}$ | $\approx 2MN$ | 2 | $\approx 2N_L MN$ | $\eta_{st}$ (3.16) |
| LAMP (Fig. 4.8) | $\mathbf{C_1}^{N\times M},\ldots,\mathbf{C_{N_L}}^{N\times M},\alpha_1,\ldots,\alpha_{N_L}$ | $\approx N_L MN$ | 2 | $\approx 2N_L MN$ | $\eta$ |
| D$l_0$RE (Fig. 4.9) | $\mathbf{C}^{N\times M},\mathbf{S}^{N\times N},\theta$ | $\approx N^2+MN$ | 1 | $\approx N_L N^2$ | $\eta_H$ (4.9) |
| TISTA (Fig. 4.11) | $\gamma_1,\ldots,\gamma_{N_L}$ | $\approx 2MN$ | 2 | $\approx 2N_L MN$ | $\eta_{MMSE}$ |
| SURE-TISTA (Fig. 4.12) | $\lambda_1,\ldots,\lambda_{N_L}$ | $\approx 2MN$ | 2 | $\approx 2N_L MN$ | $I$ (4.22) |

only related to the number of layers $N_L$ while the number of learned parameters required in LISTA, LAMP and D$l_0$RE also depends on the values of $M$ and $N$. Therefore, the number of learnable variables for TISTA and SURE-TISTA is much smaller than those of LISTA, LAMP and D$l_0$RE. It significantly reduces the training complexity of TISTA and SURE-TISTA.

If $N > 2M$, the computational complexity of LISTA (for the representation in Fig. 4.6) and D$l_0$RE is bigger than those of LISTA (for the representation in Fig. 4.7), LAMP, TISTA, and SURE-TISTA. Regarding to the memory complexity, LAMP can have the biggest one compared to the others NN presented in Table 4.2. It occurs because the memory complexity of LAMP also depends on $N_L$ while the others are only related to $M$ and $N$ values.

### 4.3.2 Neural Network with Complex Number

In communication systems, normally the channel and the signals are complex. In generally, the deep networks are not adapted to deal with complex number. Let $real(.)$ and $imag(.)$ be the real and imaginary parts of a complex vector, respectively. In order to use neural network to estimate complex sparse signals, the modification presented in (4.23) can be applied.

$$\mathbf{y}^R = \begin{pmatrix} real(\mathbf{y}) \\ imag(\mathbf{y}) \end{pmatrix}, \quad \mathbf{h}^R = \begin{pmatrix} real(\mathbf{h}) \\ imag(\mathbf{h}) \end{pmatrix},$$
$$\mathbf{n}^R = \begin{pmatrix} real(\mathbf{n}) \\ imag(\mathbf{n}) \end{pmatrix}, \quad \mathbf{A}^R = \begin{pmatrix} real(\mathbf{A}) & -imag(\mathbf{A}) \\ imag(\mathbf{A}) & real(\mathbf{A}) \end{pmatrix} \tag{4.23}$$

Therefore, (2.8) can be replaced by (4.24).

$$\mathbf{y}^R = \mathbf{A}^R\mathbf{h}^R + \mathbf{n}^R \tag{4.24}$$

### 4.3.3 Challenges

Using neural network to sparse signal recovery takes advantage of potentially available training data. However, although the use of neural network appears to perform improvements for some applications, it presents some challenges. Maybe it can be necessary to train a specific network for each SNR value. Training at a low SNR does not allow to fit the best network in higher SNR scenarios [189]. The design of the neural network architecture and the optimal choice of loss and activation functions are other difficulties related to neural networks. The quantity of the training data $\{(\mathbf{y}^d, \mathbf{h}^d)\}_{d=1}^{D}$ to be used is also important to better fit the network.

Moreover, the majority of building blocks, techniques, and architectures for deep learning are based on real-valued operations and representations [211]. However, in dealing with communication systems, complex numbers are used to represent the channel, noise, and received signal. Therefore, the neural network has to be adapted to operate on complex numbers. One alternative is presented in Section 4.3.2.

The study of the LISTA led to the proposal of two improvements for this NN, both of which increase the performance of its estimation. The first one is to use the estimate of LISTA as a first result to estimate the non-zero tap positions of a sparse signal and then to calculate their values by pseudo-inversion process (see Section 5.3). The second one is to change the shrinkage function $\eta_{st}$ used in LISTA to other functions (see Section 5.5). In addition, the study of the neural networks addressed in this chapter motivated the proposition of a new neural network that is presented in Section 5.4.

# Chapter 5

# Sparse Signal Estimation Improvement Techniques

This chapter presents the sparse signal estimation improvement techniques proposed in this thesis. In Section 5.1 a greedy algorithm called Matching Pursuit based on Least Squares (MPLS) is explained. The algorithms LS, MP, OMP, and MPLS are used to estimate two different Wideband HF channels in Section 5.2. Then, an alternative to improve LISTA estimation performance is addressed in Section 5.3. A new neural network to sparse signal estimation is discussed in Section 5.4. Section 5.5 suggests some alternative shrinkage functions to replace the traditional shrinkage function used in LISTA. Finally, Section 5.6 analyzes the performance of all techniques proposed in this thesis.

Before explaining the techniques developed in this work, some definitions will be rewritten here:

- As defined in Section 3.4, the Oracle Least Square (OLS) estimator is the LS with ideal knowledge of the sparse signal on both the number of non-zero elements and their positions. Let $\mathbf{A_s}$ be the matrix generated by $s$ columns of the matrix $\mathbf{A}$, where $s$ is the sparsity of $\mathbf{h}$. The OLS calculates the non-zero coefficients of $\hat{\mathbf{h}}$ by applying the pseudo-inversion process in the matrix $\mathbf{A_s}$.

- The system model used in this chapter is defined by (2.8). Its parameter values are reported in each section of this chapter.

- The SNR is defined by (5.1)

$$SNR = \frac{E[||\mathbf{Ah}||_2^2]}{E[||\mathbf{n}||_2^2]} \tag{5.1}$$

- In some sections below, the NMSE is used to evaluate the estimation performance. Equation (3.53) is rewritten here as (5.2) to defined NMSE.

$$NMSE = \frac{1}{N_s}\sum \frac{||\mathbf{h}-\hat{\mathbf{h}}||_2^2}{||\mathbf{h}||_2^2} \tag{5.2}$$

Moreover, in this thesis, "tap attribution" is defined as to give "1" to a tap which has a non-zero value and "0" to a tap that has a zero value. For example, let $\mathbf{h}$ be the sparse signal to be estimated, its "tap attribution" is given by $\mathbf{h}_t$. Let $\hat{\mathbf{h}}$ be the estimate and $\hat{\mathbf{h}}_t$ be its "tap attribution", the percentage of non-zero taps positions correctly found is $P_c = 4/5 = 80\%$ ( number of blue "1" in $\hat{\mathbf{h}}_t$ / number of "1" in $\mathbf{h}_t$) and the percentage of taps which had an improper attribution is $P_e = 3/20 = 15\%$ ( number of red "0" and red "1" in $\hat{\mathbf{h}}_t$ / length of $\mathbf{h}_t$). These definitions will be applied in Sections 5.3, 5.4 and 5.6.

$$\begin{aligned}
\mathbf{h} &= 0\ \ 0\ \ 0.87\ \ 0\ \ 1.4\ \ 0\ \ 0\ \ 0.2\ \ 1.2\ \ 0\ \ 0.9\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0 && (5.3)\\
\mathbf{h}_t &= 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0 && (5.4)\\
\hat{\mathbf{h}} &= 0.3\ \ 0\ \ 1.2\ \ 0\ \ 0.9\ \ 0\ \ 0.8\ \ 0.3\ \ 0\ \ 0\ \ 0.8\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0 && (5.5)\\
\hat{\mathbf{h}}_t &= 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0 && (5.6)
\end{aligned}$$

## 5.1 Matching Pursuit based on Least Squares (MPLS)

The study of the algorithms presented in Chapter 3 led to the development of a new greedy algorithm, called Matching Pursuit based on Least Squares (MPLS) [212]. The MPLS algorithm builds upon the MP algorithm to find the positions of the non-zero signal taps. Similarly to the MP algorithm, each iteration of the MPLS algorithm consists in finding the column $\mathbf{a}_{k_i} \in \mathbf{A}$ which is best aligned with the residual vector $\mathbf{b}_{i-1}$ ( $\mathbf{b}_0 = \mathbf{y}$) and the selection is performed according to:

$$k_i = \arg\max_l |\mathbf{a}_l{}^H \mathbf{b_{i-1}}|, \quad l = 1,2,....,N \tag{5.7}$$

The difference between the MP and the MPLS algorithms lies in how the values assigned to the non-zero signal taps are calculated. The MP algorithm estimates the signal as the projection values, while the MPLS algorithm does this estimation through the LS algorithm in the end of the algorithm.

Consider $\Lambda_i$ the index set of the best aligned columns of $\mathbf{A}$ until the iteration $i$, $\Lambda_i$ is updated by:

- If $k_i \notin \Lambda_{i-1}$, the index set is updated as $\Lambda_i = \Lambda_{i-1} \cup k_i$.
- Otherwise, $\Lambda_i = \Lambda_{i-1}$.

Then, the residual vector's projection along this direction is removed and a new residual vector is computed as (5.8).

$$\mathbf{b}_i = \mathbf{b}_{i-1} - P_{\mathbf{a}_{k_i}} \mathbf{b}_{i-1} = \mathbf{b}_{i-1} - \frac{(\mathbf{a}_{k_i}{}^H \mathbf{b}_{i-1})\mathbf{a}_{k_i}}{||\mathbf{a}_{k_i}||_2^2} \tag{5.8}$$

After reaching the stop criterion, the signal is estimated by (5.9), where $T$ is the number of iterations and $\mathbf{A}(\Lambda_T)$ is a submatrix of $\mathbf{A}$ consisting of the $\mathbf{a}_i$ columns with $i \in \Lambda_T$.

$$\hat{\mathbf{h}} = \mathbf{A}^\dagger(\Lambda_T)\mathbf{y} \tag{5.9}$$

### 5.1.1 Application Case

The performance of MP, MPLS and OMP algorithms for sparse channel estimation were evaluated and compared to the theoretical performance bound OLS.

The Mean-Square Deviation (MSD) described by (5.10) is used to evaluate the algorithms in terms of Sparsity ($S$) variation, where $N_s$ is the number of channel simulations.

$$MSD = \frac{1}{N_s} \sum ||\mathbf{h} - \hat{\mathbf{h}}||_2^2 \tag{5.10}$$

The system model is defined by (2.8), considering:

- $N = 120$.
- $SNR = 10\,\text{dB}$.
- $\mathbf{A}$ stands for the $M \times N$ measurement matrix. The $i$th column of $\mathbf{A}$ is denoted $\mathbf{a}_i$. $\mathbf{A}$ is a Toeplitz matrix determined by the training sequence $\mathbf{c} = [c_1, c_2, ..., c_M]^T$ where $c_i$ is a QPSK value.

- Each element of the sparse signal $h$ to be estimated is characterized by $h_i = s_i d_i$. The complex-valued vector $\mathbf{d} = [d_1, d_2, ..., d_N]^T$ characterizes the path gains as a complex gaussian distribution, while the binary vector $\mathbf{s} = [s_1, s_2, ..., s_N]^T$ describes the channel paths. Each $s_i \in \{0, 1\}$ denotes whether there is a non-zero channel tap at index $i$. The elements of the vector **s** correspond to the combination of the possible positions of the non-zero elements in the vector **h**, chosen randomly and with a number of elements defined by the sparsity of the channel.

- **n** is the Additive White Gaussian Noise (AWGN).

### 5.1.2 Estimation Performance

Comparisons of sparse channel estimation algorithms reported in the literature take into account only one channel with its fixed position of non-zero channel taps [20, 25, 26]. More generally, performance analyses reflect only a fixed scenario. To obtain better precision, $N_s = 5000$ sparse channels with length $N = 120$ were simulated for each point. Furthermore, performance analysis of the proposed algorithm are performed under a variety of scenarios by varying the sparsity of the channel, the length of the training sequence and the stopping criterion.

Fig. 5.1 presents the LS, MP, MPLS, OMP and OLS algorithms performances in terms of MSD versus channel sparsity for $M = 60$. Let $N_{it}$ be the number of iterations (stopping criterion), in Fig. 5.1, the continuous lines represent the simulations with $N_{it} = 2S$, while the broken lines denote the simulations with $N_{it} = S$.

When $N_{it} = 2S$, the MPLS and the OMP algorithms have similar results in terms of MSD until 16 non-zero channel taps, but it should be noted that the MPLS algorithm achieves these results with lower complexity than the OMP algorithm. For $S < 10$, MP, OMP, and MPLS algorithms lead to similar results. Therefore, for this range of $S$ values, MP algorithm is preferable due to its lower complexity.

Notice that, for small values of $S$ and $N_{it} = S$, the MSD of the OMP and MPLS algorithms are close to the one achieved by OLS. This means that both algorithms find almost all correct positions of the non-zero channel taps. Moreover, comparing the results for $N_{it} = 2S$ and $N_{it} = S$, it can be seen that when the stop criterion is exactly the number of non-zero channel taps ($N_{it} = S$), the MSD of both OMP and MPLS algorithms decrease unlike what happens with the MP algorithm.

Given that the MP and the MPLS algorithms choose the non-zero taps positions in the same way, different MSD values obtained are likely due to the value assigned to the non-zero channel

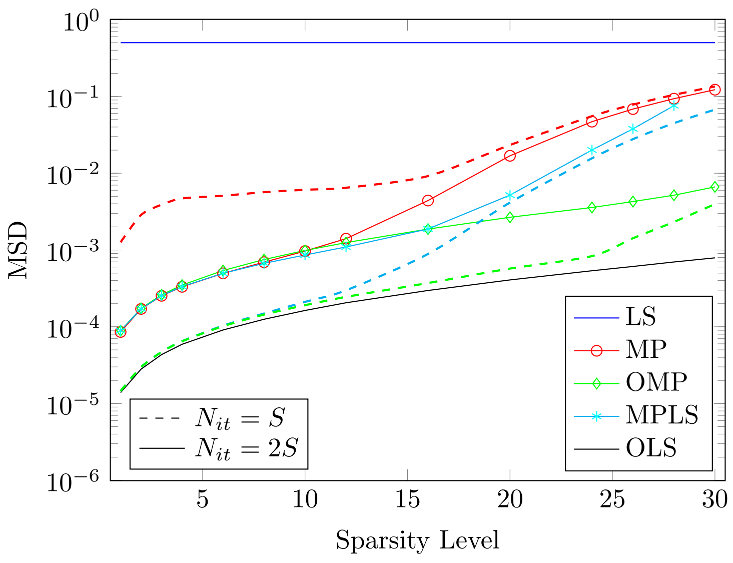


Figure 5.1 – MSD vs Sparsity for $SNR = 10dB$, $M = 60$.

taps. On the other hand, assigning the projection value to the channel estimation value (which is done by the MP algorithm) has a very damaging effect on channel estimation. This does not occur with MPLS algorithm, since it finds the correct positions of the non-zero channel taps and estimates their values using the LS algorithm, thus approaching to the results of the OLS estimation.

Table 5.1 shows the amount of channels for which the algorithms found the correct positions of non-zero channel taps considering stopping criteria $SC$, $N_s = 5000$, $S = 6$, and $SNR = 10\,\text{dB}$. This table reinforces the conclusions from Fig. 5.1. Indeed, even if the MP algorithm finds the non-zero channel taps properly, it leads to higher MSD value compared to those from the MPLS and the OMP.

Table 5.1 – Number of channels that found the correct positions of non-zero channel taps.

| | 5 correct positions | | 6 correct positions | |
|---|---|---|---|---|
| SC | MP/MPLS | OMP | MP/MPLS | OMP |
| $N_{it} = S$ | 499 | 480 | 4484 | 4503 |
| $N_{it} = 2S$ | 401 | 386 | 4589 | 4605 |

Fig. 5.2 considers $M = 120$, i. e., the same length of the channel. The stopping criterion is the number of iterations equivalent to the number of non-zero channel taps (i.e., $N_{it} = S$). Comparing Fig. 5.1 and Fig. 5.2, it can be seen that as $M$ increases, the MSD of the algorithms decreases.

It is worth mentioning that for small values $S$, a smaller $M$ value can be used with the OMP and the MPLS algorithms without greatly increasing the MSD value.

These results points out an advantage of using algorithms that take into account the sparsity

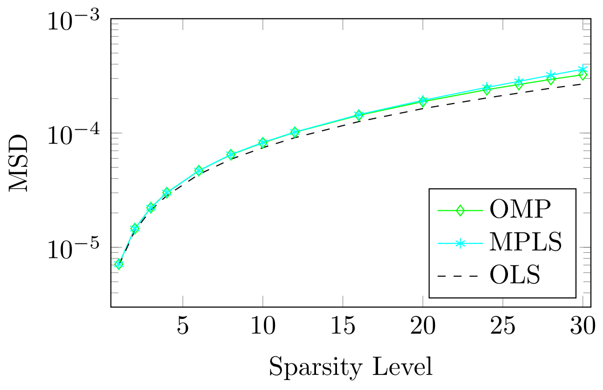


Figure 5.2 – MSD vs Sparsity for $SNR = 10dB$, stopping criterion $N_{it} = S$, $M = 120$.

of the channel. It is noted that the OMP and the MPLS algorithms have similar results in terms of MSD, with the advantage that the MPLS algorithm is less computationally demanding.

## 5.2 Application of the CS on Wideband HF Channel Estimation

This section analyses the Wideband HF Channel Estimation using the algorithms MPLS, MP and OMP.

The High Frequency (HF) band ranges from 3 to 30 MHz and provides Beyond Line Of Sight and long-distance (often intercontinental) communications without the use of satellites or radio-relay, with low operational costs in comparison to fixed or satellite communication systems. HF radio communications are widely used for civilian and military applications on long-distance communications, mainly in remote regions and in emergency scenarios such as natural disasters [213].

HF channel models are characterized by several parameters related to the behavior of the ionosphere. In general, a stochastic behavior that represents the different possible ways for the transmission to reach the receiver is modeled. Given the HF transmission characteristics, sometimes it is possible to establish transmissions from some tens of kilometers near vertical incidence skywave (NVIS) until distances far over-the-horizon (OTH) of the order of tens of thousands of kilometers [214]. This large variety of scenarios brings great differences, for example, in delay scattering of the received signal, which could be of an order of 10 ms [215].

Several standards have been developed to standardize HF communications [216, 217]. In traditional systems, the bandwidth of HF communications is very narrow (3 kHz) and the transmission rate is very low, about 600 bps or 2400 bps [217]. However, with the increasing amount of data transmission, it is necessary to increase the transmission rates, which requires a bigger bandwidth of HF communications. For example, the MIL-STD-188-110C in its Appendix D establishes the characteristics of the waveforms for transmission in wideband HF channels with bandwidth up to 24 kHz and data rates ranging from 75 bps for the lowest 3 kHz rate to 120 kbps for the highest 24 kHz rate [217]. The wideband HF channel is also used in spread spectrum communication, that is, in situations where stealthy and robust communications are required [218].

The HF channel is a time-varying fading channel, involving time delay, Doppler frequency shift, interference, and noise. In a HF channel, the transmitted signal travels over several paths with different delays to the receiver via single or multipath reflects from the ionosphere (see Fig. 5.3). This channel is characterized by multipath propagation and fading as in (5.11) where $h_l(t,\tau)$ is the impulse response for one of different ionospheric propagation paths (or reflecting ionospheric layers); $l$, $t$, and $\tau$ are independent variables for time and delay [219].

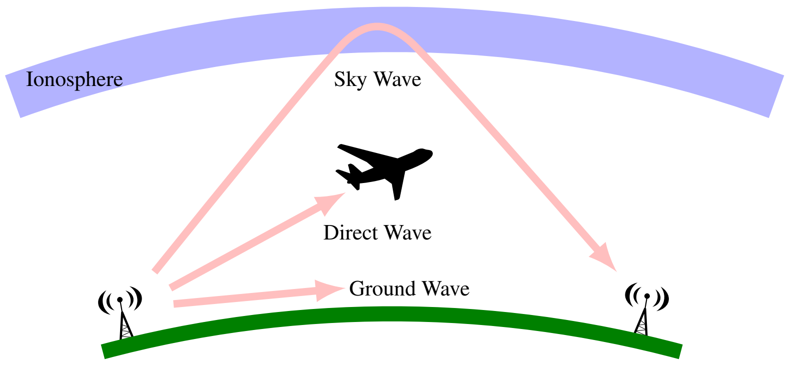


Figure 5.3 – Types of radio propagation: sky wave (HF), direct wave and ground wave.

$$h(t,\tau) = \sum_{l=0}^{L-1} h_l(t)\delta(\tau - \tau_l) = \sum_{l=0}^{L-1} h_l(t,\tau) \tag{5.11}$$

The basic concepts of wideband HF channels are investigated in [220]. Some laboratories have conducted several wideband HF propagation measurement experiments useful in the channel model development process [221, 222]. A wideband HF simulation system is described in [219], the wideband HF channel model proposed is more appropriate than the Watterson channel model for representing wideband HF channels.

The Watterson channel model [223] neglects the time delay spread and the time delay of each path has a single value. Hence, it can not be used as a wideband HF channel model, because the effects of delay time spread, Doppler frequency shift, Doppler spread and variation of Doppler shift with delay must be described in the wideband HF channel model. To address these characteristics, for each propagation path, the model of the impulse response is given by [219]:

$$h_l(t,\tau) = \sqrt{P_l(\tau)} D_l(t,\tau) \Psi_l(t,\tau) \tag{5.12}$$

where $P_l(\tau)$ is the delay power profile (DPP) that determines how the impulse response behaves, its square root $\sqrt{P_l(\tau)}$ describes the shape in delay dimension, $D_l(t,\tau)$ is the deterministic phase function describing the Doppler shift of each path, and $\Psi_l(t,\tau)$ is the stochastic modulation function [219].

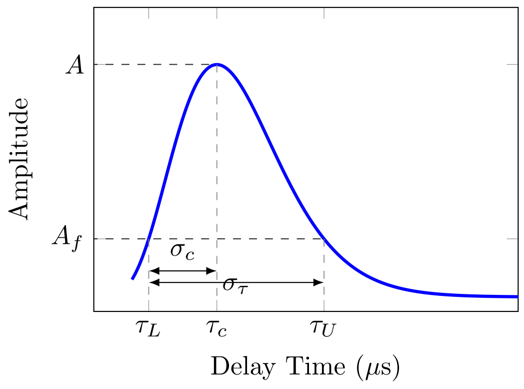


Figure 5.4 – Delay Power Profile.

Fig. 5.4 specifies the DPP, where $\sigma_\tau$ is the delay spread, $\sigma_c$ is the rise time, $A$ is the maximum value of the DPP. Let $\tau$ be the delay variable, $\Delta$ controls the width, $\tau_c$ controls the delay offset, $\alpha$ controls the symmetry of the profile, and $\Gamma(.)$ is the Gamma function, $P_l(\tau)$ is defined by [219]:

$$P_l(\tau) = A\frac{\alpha^{\alpha+1}}{\Delta\Gamma(\alpha+1)} z^{\alpha} e^{-\alpha z} \tag{5.13}$$

$$z = \frac{\tau-\tau_c}{\Delta} + 1 > 0 \tag{5.14}$$

Let $f_S$ be the Doppler shift at $\tau_c$ and $b$ be the rate of change of the Doppler shift between $\tau_L$ and $\tau_c$, the deterministic phase function $D_l(t,\tau)$ can be given by [219]:

$$D_l(t,\tau) = e^{i2\pi[f_S+b(\tau-\tau_c)]t} \tag{5.15}$$

The stochastic modulation function $\Psi_l(t,\tau)$ describes the fading of the impulse response. $\Psi_l(t,\tau)$ is independent in delay $\tau$, correlated in time $t$. It is defined in terms of its resultant Doppler spread width and spectral shape [219]:

$$\Psi_t = x_t + iy_t \quad (t = 0, 1, 2, ...) \tag{5.16}$$

where $x_t$ and $y_t$ are the independent random variables generated by independent random variables $\rho_t$ and $\rho_t'$ [224]:

$$x_t = \rho_t + (x_{t-1} - \rho_t)\lambda \tag{5.17}$$

$$y_t = \rho_t' + (y_{t-1} - \rho_t')\lambda \tag{5.18}$$

$$x_0 = (1 - \lambda)\rho_0 \tag{5.19}$$

$$y_0 = (1 - \lambda)\rho_0' \tag{5.20}$$

$$\lambda = exp[-(\Delta t)\sigma_f] \tag{5.21}$$

$\Psi_l(t, \tau)$ can be generated as random variables having an autocorrelation function with Gaussian shape or Lorentzian shape [219, 224].

For example, Fig. 5.5 shows a wideband HF channel impulse response with 4 propagation paths and delay spread $\sigma_\tau = 100\mu$s. The symbol rate is 19 200 symbol/s, that is used for a bandwidth of 24 kHz according to the Appendix D of the MIL-STD-188-110C [217].

Since the HF transmissions have a small number of arriving rays, the HF channel can be seen as a tapped delay line in the delay spread domain. This section investigates the wideband HF channel estimation using sparse recovery algorithms.

In order to do this, the system model is defined by (2.8), where $\mathbf{y} = [y_1, y_2, ..., y_M]^T$ is the received signal, $\mathbf{h} = [h_1, h_2, ..., h_N]^T$ denotes the time-domain discrete wideband HF channel vector with $N > M$ taps and $\mathbf{n}$ is the Additive White Gaussian Noise (AWGN). $\mathbf{A}$ stands for the $M \times N$ measurement matrix. The $i$th column of $\mathbf{A}$ is denoted $\mathbf{a}_i$. $\mathbf{A}$ is a Toeplitz matrix determined by the QPSK training symbols.

Two different wideband HF channels are considered and the Mean-Square Deviation (MSD) described by (5.10) is used to evaluate the algorithms in terms of the quantity of the acquired measurement ($M$) and the number of iterations ($N_{it}$), where $N_s$ is the number of channel simulations. For these simulations, $N = 127$ and $SNR = 20$ dB.

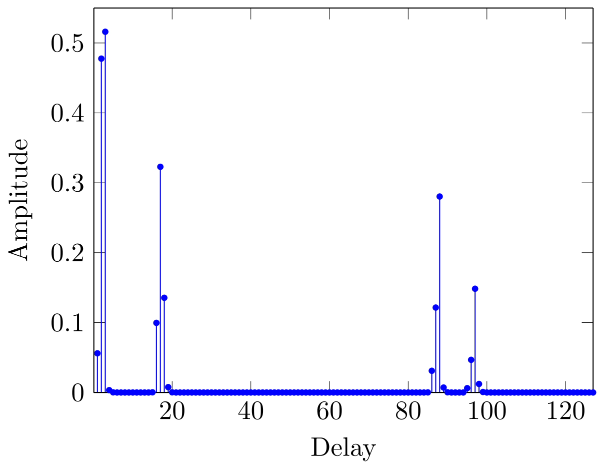


Figure 5.5 – Example of a wideband HF channel with 4 propagation paths.

### 5.2.1 Wideband HF Channel A

The wideband HF Channel A is a case of near vertical incidence skywave (NVIS). It is a radio propagation mode which involves the use of antennas with a very high radiation angle. Channel A has 2 propagation paths with delay spread of $10\mu$s and $200\mu$s separated by 0.8 ms. These channel model parameters adjust the scattering function of the simulated channel to the scattering function of the reported measured channel presented in [225].

The comparison of the LS, the MP, the OMP, and the MPLS algorithms in terms of MSD performance versus the number of iterations ($N_{it}$) is shown in Fig. 5.6. The quantity of the acquired measurement is $M = 40$.

It can be noticed that after some iterations, the MSD value of the MP, the MPLS, and the OMP algorithms have increased with each new iteration. It occurs because these algorithms do not know the exact channel's sparsity ($S$) neither the exactly $S$ non-zero tap positions. In fact, the wideband HF Channel A has few non-zero taps, so when these algorithms choose more non-zero taps, they are choosing tap positions with almost only the noise influence or re-selecting (it can occur with the MP and the MPLS algorithms) a tap and increasing the noise influence in this tap.

The LS algorithm gives the worst channel estimate because it does not consider the channel sparsity in its estimations. Then, the bad estimation influence is given in almost all taps for the channel estimate. Therefore, it is possible to see that if the MP algorithm is used, the MSD is lower. This occurs because this algorithm uses the sparsity characteristic of the channel to better estimate it. Moreover, if MP selects more non-zero tap positions than the sparsity $S$ of the signal to be estimated, MP attributes a small value to these positions. It differs from the MPLS that uses the

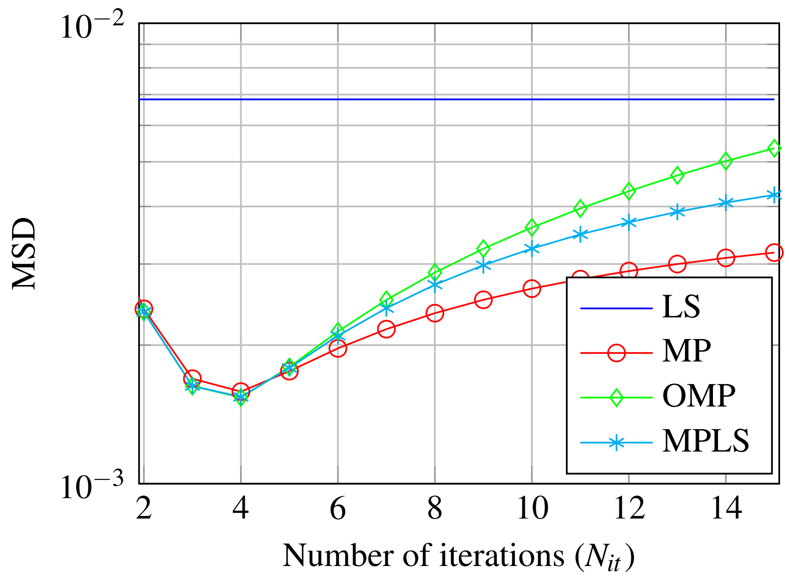


Figure 5.6 – MSD vs $N_{it}$ for $M = 40$ and wideband HF Channel A.

pseudo-inversion process to calculate its values and the OMP that in each new iteration chooses a new non-zero position, so when $N_{it} > S$, the OMP is necessarily increasing the MSD value.

### 5.2.2 Wideband HF Channel B

The wideband HF Channel B has 4 propagation paths and each path has a delay spread of $100\mu$s. One example is showed in Fig. 5.5. To obtain better precision, simulations in this work consider 5000 wideband HF channels where the delay position of each path was chosen randomly in each simulation.

The comparison of the LS, the MP, the OMP, and the MPLS algorithms in terms of MSD performance versus the number of iterations ($N_{it}$) is shown in Fig. 5.7. The quantity of the acquired measurement is $M = 40$.

It can be seen that for $S < 6$, the MP, the MPLS, and the OMP algorithms have similar results in terms of MSD, but for $S > 6$, the MP algorithm achieves a lower MSD value than the MPLS and the OMP algorithms. It can be explained due to the bad estimation influence in the taps which would be zero or almost zero. Besides this, the MPLS and the OMP algorithms use the LS solution to estimate the tap values. Then, the bad estimation influence goes to almost all taps. Otherwise, the MP algorithm gives the bad estimation influence from a bad tap selected only in this tap value estimate.

Fig. 5.8 presents MSD values varying the quantity of the acquired measurement for $S = 4$. It

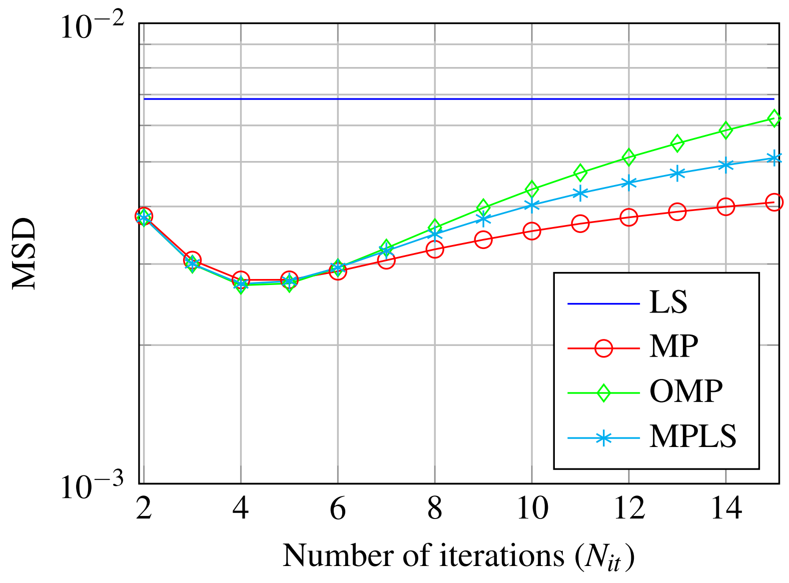


Figure 5.7 – MSD vs $N_{it}$ for $M = 40$ and wideband HF Channel B.

can be noticed that when $M$ increases, the MSD value for the algorithms analyzed decreases. It is worth mentioning that a smaller quantity of the acquired measurement, for example $M = 40$, can be used with the algorithms and achieve a low MSD value. This points out an advantage of using recovery algorithms that take into account the channel's sparsity for wideband HF channel estimation.

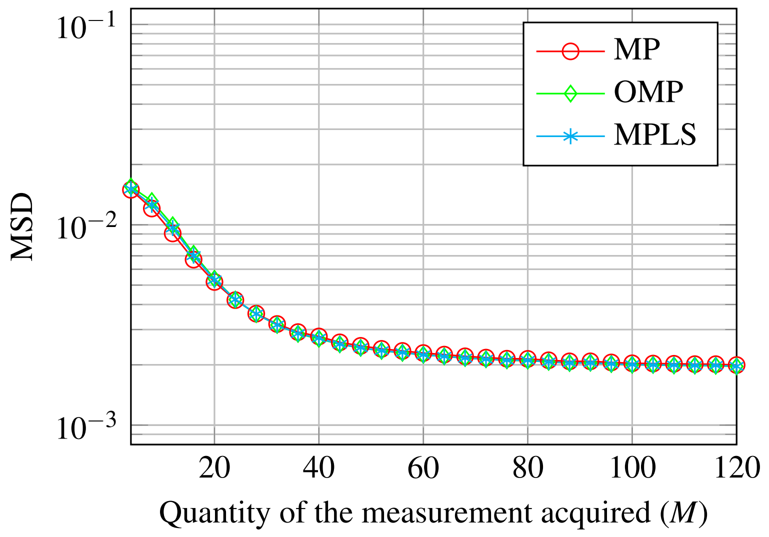


Figure 5.8 – $MSD$ vs $M$ for $S = 4$ and Wideband HF Channel B.

The results suggest that wideband HF channels could be considered sparse in the delay spread domain. Simulations have shown that the use of a sparse recovery algorithm that considers the channel's sparsity achieves a better estimation performance than others that do not take into account the channel's sparsity.

## 5.3 Proposition to Improve the LISTA Performance

This section discusses a solution to improve the LISTA estimation performance. Fig. 5.9 shows an example of a sparse signal $\mathbf{h}$ and its estimate $\hat{\mathbf{h}}$ given by $N_L = 20$ layers of LISTA [226]. As can be seen, LISTA can estimate almost all the non-zero tap positions relatively well. However, LISTA improperly attributes non-zero values to several zero-value taps, so negatively impacting the estimation accuracy.

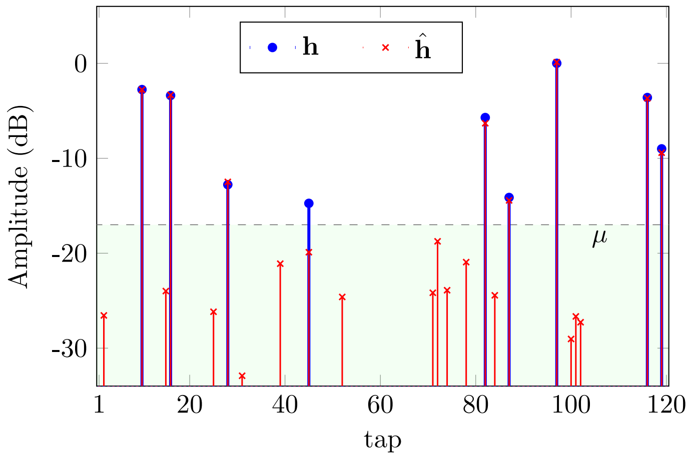


Figure 5.9 – Example of a sparse signal $\mathbf{h}$ and its estimate $\hat{\mathbf{h}}$ given by $N_L = 20$ layers of LISTA.

From Fig. 5.9, it can be noticed that using an appropriate threshold $\mu$ allows to avoid undue non-zero values. With this in mind, Fig. 5.10 shows the general schema of the proposed sparse recovery approach which is called in this thesis as "Tech. 1" [226].

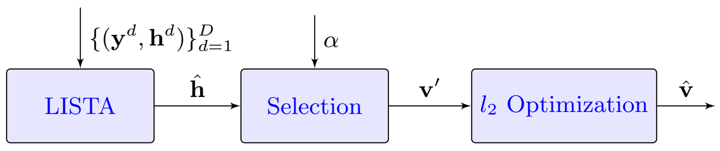


Figure 5.10 – General schema of "Tech. 1".

The first step consists of LISTA implementation, which produces the estimate $\hat{\mathbf{h}}$. The second step ("Selection") takes into account $\alpha$ and $\hat{\mathbf{h}}$ to produce the vector $\mathbf{v}'$. This step aims to use a threshold $\mu$ as presented in Fig. 5.9. Let $g$ be the number of non-zero taps of $\hat{\mathbf{h}}$, $E$ be the biggest absolute tap value of $\hat{\mathbf{h}}$ and $\alpha$ be a positive value, the elements of the vector $\mathbf{v}'$ are calculated by:

$$\begin{aligned} v'(i) &= 1, \quad if \quad \hat{h}(i) > \mu \\ v'(i) &= 0, \quad if \quad \hat{h}(i) \leq \mu \end{aligned} \tag{5.22}$$

where $\mu = \alpha E$.

That is, this step generates a signal $\mathbf{v}'$ with $j$ elements 1 where $j \leq g \leq N$. The positions of these $j$ non-zero elements will be the same positions of the non-zero elements of $\hat{\mathbf{v}}$.

Let $\mathbf{G}(\mathbf{x})$ be the matrix $M \times K$ composed by only $K$ columns of the matrix $\mathbf{A}$ correlated to the positions of the non-zero elements of $\mathbf{x}$. The function $U(\mathbf{x})$ is defined by:

$$U(\mathbf{x}) = \mathbf{G}^{\dagger}(\mathbf{x})\mathbf{y} = \mathbf{G}^{T}(\mathbf{x})(\mathbf{G}(\mathbf{x})\mathbf{G}^{T}(\mathbf{x}))^{-1}\mathbf{y} \tag{5.23}$$

Finally, step "$l_2$ Optimization" calculates the non-zero elements of $\hat{\mathbf{v}}$ by $U(\mathbf{v}')$. Then, the estimate $\hat{\mathbf{v}}$ of the sparse signal $\mathbf{h}$ is calculated by attributing the values of the vector generated by $U(\mathbf{v}')$ into the positions $i$ where $v'(i) = 1$.

### 5.3.1 Application Case

The normalized mean squared error (NMSE) described by (5.2) is used to evaluate the proposition of improvement of LISTA performance ("Tech. 1"). $N_s$ is the number of realizations, which corresponds to the number of different $\mathbf{h}$ to be estimated. In this work, $N_s = 10000$.

The proposed sparse recovery approach considers the system model defined in (2.8), where:

- $N = 120$.
- $M = 60$.
- $\mathbf{A}$ is i.i.d. Gaussian, with $\mathcal{N}(0, M^{-1})$.
- $\mathbf{h}$ is Bernoulli-Gaussian. Its elements are i.i.d $\mathcal{N}(0,1)$ with probability $\gamma = 0.1$ and the others are set to 0.
- SNR = 30 dB.

The parameters of LISTA were directly learned from independent realizations of the training data during the training phase. Training data sets $\{(\mathbf{y}^d, \mathbf{h}^d)\}_{d=1}^{D}$ with $D = 10000$ were used. Another set with a size of 10000 was used for the testing phase. It was generated independently of the training data but from the same distribution. During the training phase, LISTA adapts the parameters minimizing the loss function (4.3) using the Adam optimizer [197].

### 5.3.2 Estimation Performance

Fig. 5.11 presents the obtained NMSE as a function of the LISTA's number of layers $N_L$. The curve LISTA$_{\text{std}}$ (in blue) represents the NMSE values considering $\hat{\mathbf{h}}$ as the estimate of the sparse signal $\mathbf{h}$, that is, the output of the first step in Fig. 5.10 [226]. The curve in which $\alpha = 0.0$ considers the positions of the non-zero taps of $\hat{\mathbf{h}}$ as being those of non-zero taps of the $\hat{\mathbf{v}}$ $(g = j)$. However, the non-zero coefficients $\hat{\mathbf{v}}$ differ from those of $\hat{\mathbf{h}}$, they are calculated by (5.23). The other curves stand for NMSE values considering different values of $\alpha$ used in the step "Selection". Fig. 5.11 also presents the theoretical performance bound Oracle Least Square (OLS).

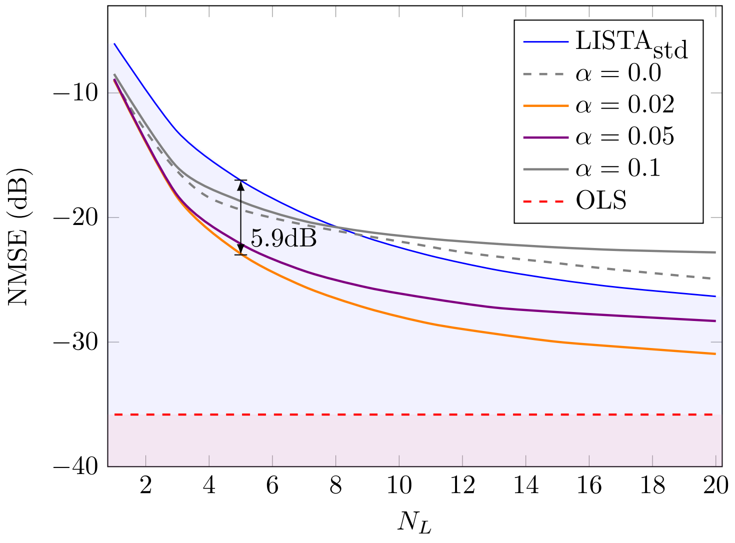


Figure 5.11 – Estimation performance for different values of $\alpha$ using "Tech. 1".

As expected, increasing the number of layers makes the NMSE values decrease. Furthermore, from Fig. 5.11, it can be seen that using the steps proposed in Fig. 5.10 the estimation performance can be improved (NMSE values decrease) [226].

As it is showed in Fig. 5.11, with $\alpha = 0.02$, the NMSE value can decrease up to 5.9 dB compared to LISTA$_{\text{std}}$. This occurs because, even if LISTA chooses good parameters $\Theta$, the estimate $\hat{\mathbf{h}}$ given by LISTA has more non-zero taps than $\mathbf{h}$, leading to worse NMSE values. Therefore, after the step "Selection", the number of the non-zero tap positions decreases and the step "$l_2$ Optmization" calculates their values using LS algorithm resulting in lower NMSE values. This conclusion is reinforced by Fig. 5.12 [226].

Fig. 5.12 presents the percentage of taps which had an improper tap attribution, that is, the algorithm attributes a non-zero value to a tap that should have a zero value or the algorithm attributes a zero value to a tap which should have a non-zero value. It can be observed that only using LISTA (curve LISTA$_{std}$), 10% to 15% of the tap attributions are incorrect. On the other hand, with $\alpha = 0.02$ this percentage of error decreases and this decrease reflects in better NMSE values as can be seen in Fig. 5.11. This occurs because in the "Selection" step, taps with a norm value lower or equal to $0.02E$ are setting to a zero value removing some improper tap attributions.

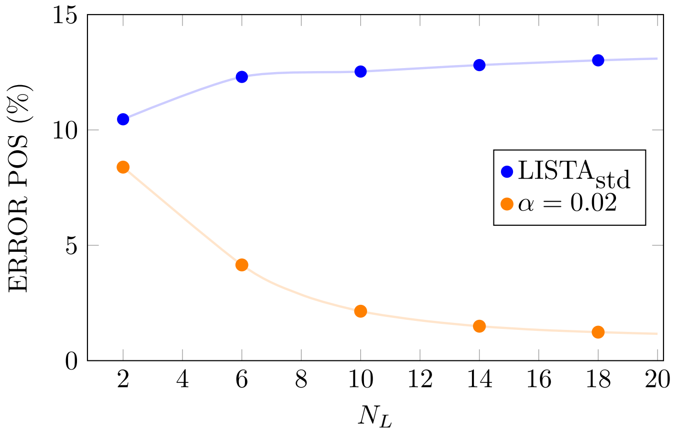


Figure 5.12 – Percentage of taps that had an improper attribution.

Moreover, it can be observed from Fig. 5.11 that the $\alpha$ value influences a lot the performance of the estimation. A big value of $\alpha$ (for example $\alpha = 0.1$) can lead to set a lot of taps to zero resulting in an estimate $\hat{\mathbf{v}}$ with sparsity $s$ lower than $\mathbf{h}$, so generating bigger NMSE values. On the other hand, a small $\alpha$ value (for example $\alpha = 0.0$) may consider that a lot of taps have non-zero values resulting in an estimate with sparsity $s$ bigger than $\mathbf{h}$.

The influence of $\alpha$ value can be also observed in Table 5.2 [226]. This table shows the quantity of layers required to achieve NMSE values equal to $-20$ dB and $-30$ dB. It considers the case without using the steps proposed in Fig. 5.10 (Case 1) and the cases using them with different $\alpha$ values (Cases 2 to 5).

Table 5.2 – Number of layers required to achieve a given performance using "Tech. 1".

| Case | Method | NMSE $= -20$ dB | NMSE $= -30$ dB |
|---|---|---|---|
| 1 | LISTA$_{\text{std}}$ | 8 | >20 |
| 2 | $\alpha = 0.0$ | 8 | >20 |
| 3 | $\alpha = 0.02$ | 4 | 16 |
| 4 | $\alpha = 0.05$ | 5 | >20 |
| 5 | $\alpha = 0.1$ | 7 | >20 |

Case 3 shows that with 16 layers and $\alpha = 0.02$ the NMSE is $-30$ dB. This value is not achieved even with 20 layers in the case that only the LISTA is used (Case 1). On the other hand, less adjusted values of $\alpha$ may require bigger values of $N_L$ (>20) to achieve NMSE close to $-30$ dB.

Furthermore, it can be noticed that to achieve NMSE $= -20\,\text{dB}$ only 4 layers are required in Case 3. Nevertheless, 8 layers are required if the steps proposed in Fig. 5.10 aren't applied (Case 1), so increasing the training phase time and also increasing the number of matrix-vector multiplications required in the evaluation phase.

These results show that applying the two steps proposed after the LISTA output may lead to better estimations (lower NMSE values) and fewer layers of the neural network than directly consider the output given by LISTA as the estimate of the signal of interest. However, even if these steps better estimate the sparse signal, the threshold used can set a zero value to a tap that should have a non-zero value. With this in mind, a new neural network is proposed and described in Section 5.4 to improve the sparse signal estimation focuses on finding the non-zero tap positions.

The steps proposed in Fig. 5.10 can be applied in other neural networks. For instance, it is possible to change the first step "LISTA" for another neural network for example LAMP and to use the estimate given by LAMP as $\hat{\mathbf{h}}$ in Fig. 5.10.

## 5.4 Proposed Neural Network

This section describes the proposed neural network to sparse signal estimation. Fig. 5.13 illustrates the proposed neural network with $N_L$ layers. In this thesis, this network is also called "Tech. 2" [226]. The aim of this network is to estimate the non-zero tap positions of the sparse signal instead of directly estimate its elements. Once the non-zero tap positions is estimated by the proposed network, their element values are computed solving the least square problem. The output $\hat{\mathbf{p}}_i$ of the $i$-layer of the proposed network is given by:

$$\hat{\mathbf{p}}_i = \eta_r\left(\mathbf{S}\hat{\mathbf{p}}_{i-1} + \mathbf{C}\mathbf{y}; \lambda_i\right) \tag{5.24}$$

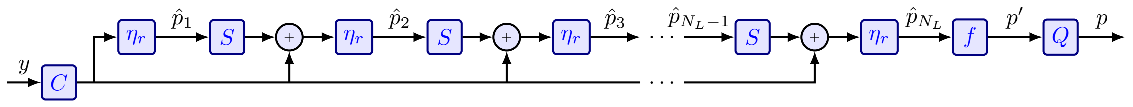


Figure 5.13 – Proposed neural network with $N_L$ layers.

Similarly to LISTA, the matrices $\mathbf{C}$ and $\mathbf{S}$, and the thresholds $\lambda_i$ are "learned" during the training phase. However, while in LISTA the training data is composed by sparse signals $\mathbf{h}^d$, the training data in the proposed neural network is $\{(\mathbf{y}^d, \mathbf{t}^d)\}_{d=1}^{D}$ where $t^d(j) = 1$ if $h^d(j) \neq 0$ and $t^d(j) = 0$ otherwise. The loss function used in the proposed neural network is given by (5.25) where $\Theta = [\mathbf{C}, \mathbf{S}, \lambda]$ and $\lambda = [\lambda_1, \lambda_2, ..., \lambda_{N_L}]$.

$$L(\Theta) = \frac{1}{D} \sum_{d=1}^{D} ||\mathbf{p}'(\mathbf{y}^d; \Theta) - \mathbf{t}^d||_2^2 \tag{5.25}$$

In this thesis, two different $\eta_r$ functions are used: $\eta_{st}$ and $\eta_{pwl}$. The first one, $\eta_r = \eta_{st}$, is the same function used in LISTA and defined by (3.16). On the other hand, $\eta_r = \eta_{pwl}$ is defined by (5.26), where $\theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$ [227].

$$\eta_{pwl}(x;\theta) = \begin{cases} \theta_3 x, & if \quad |x| \leq \theta_1 \\ sgn(x)[\theta_4(|x| - \theta_1) + \theta_3\theta_1], & if \quad \theta_1 < |x| \leq \theta_2 \\ sgn(x)[\theta_5(|x| - \theta_2) + \theta_4(\theta_2 - \theta_1) + \theta_3\theta_1], & if \quad \theta_2 < |x| \end{cases} \tag{5.26}$$

After the last layer of the proposed neural network, two more steps are performed (see Fig. 5.13):

- $f$: it is the bounded function defined by (5.27) and illustrated in Fig. 5.14. This function was chosen because it is a symmetric function and it generates only values $\in [0,1]$ resulting in $p'(i) \in [0,1]$.

$$f(x) = |tanh(x)| = |\frac{e^x - e^{-x}}{e^x + e^{-x}}| \tag{5.27}$$

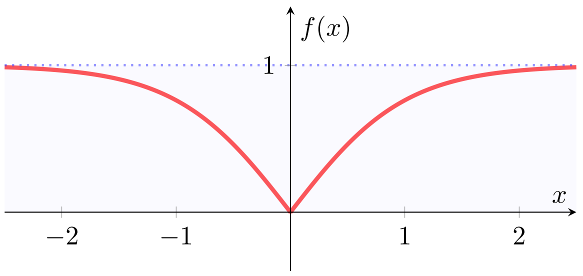


Figure 5.14 – $f(x) = |tanh(x)|$.

- $Q$: the vector $\mathbf{p}'$ is transformed into the vector $\mathbf{p}$ composed by only elements 0 and 1. Let $\tau \in [0,1]$, so:

$$\begin{aligned} p(i) = 1, \quad if \quad p'(i) \geq \tau \\ p(i) = 0, \quad if \quad p'(i) < \tau \end{aligned} \tag{5.28}$$

After the proposed neural network, $\mathbf{p}$ is the estimate of the non-zero tap positions of $\hat{\mathbf{h}}$. In other words, $p(i) = 0$ corresponds to the taps of $\hat{\mathbf{h}}$ which have a 0 value and $p(i) = 1$ represents the taps of $\hat{\mathbf{h}}$ which have a non-zero value.

The non-zero elements of $\hat{\mathbf{h}}$ are calculated through the least square solution. In other words, the non-zero elements of the estimate $\hat{\mathbf{h}}$ of the sparse signal $\mathbf{h}$ are calculated by $U(\mathbf{p})$ (see (5.23)) and compose the vector $\mathbf{u}$. Then, the estimate $\hat{\mathbf{h}}$ is calculated by attributing the values of the vector $\mathbf{u}$ into the tap positions of the non-zero elements of $\mathbf{p}$.

The major differences between LISTA and the proposed neural network are [226]:

- While LISTA directly estimates $\hat{\mathbf{h}}$, the proposed neural network estimates the tap positions of the non-zero elements of $\hat{\mathbf{h}}$.
- In order to estimate the non-zero tap positions, two other "steps" are introduced after the last layer of the network (see Fig. 5.13).
- The element values of $\hat{\mathbf{h}}$ aren't directly calculated by the proposed neural network. They are calculated through the least square solution based on the proposed neural network output.
- The training data is different. In LISTA it is composed by sparse signals $\mathbf{h}^d$ with different element values, while in the proposed neural network it is composed by the non-zero position vector $\mathbf{t}^d$, that is, $\mathbf{t}^d$ has only 0 or 1 as element values.

Concerning the training phase, the parameters are learned in two steps. Firstly, only the parameter $\lambda_i$ of layer $i$ is learned. Then a global learning is performed, that is, all the parameters ($C$, $S$, $\lambda_1$, $\lambda_2$, ..., $\lambda_i$) are re-learned.

### 5.4.1 Application Case

This section presents an application case of the proposed neural network. The NMSE described by (5.2) is used to evaluate the proposed neural network performance in terms of the number of layers ($N_L$). Moreover, the percentage of improper "tap attribution" is also analyzed (see definition in Section 5.3.2).

The system model parameters are the same described in Section 5.3.1 and $\tau = 0.2$. Training data sets $\{(\mathbf{y}^d, \mathbf{t}^d)\}_{d=1}^{D}$ with $D = 10000$ were used. An independent test set with a size of 10000 from the same distribution are also used. Finally, the parameters of the network were calculated minimizing the loss function described by (5.25) using the Adam optimizer.

### 5.4.2 Estimation Performance

The curves Tech. 1 and LISTA$_{std}$ showed in Fig. 5.15 and Fig. 5.16 are the same as those presented in Fig. 5.12 and Fig. 5.11, respectively [226]. They were reproduced here to better compare the results achieved by the proposed neural network.

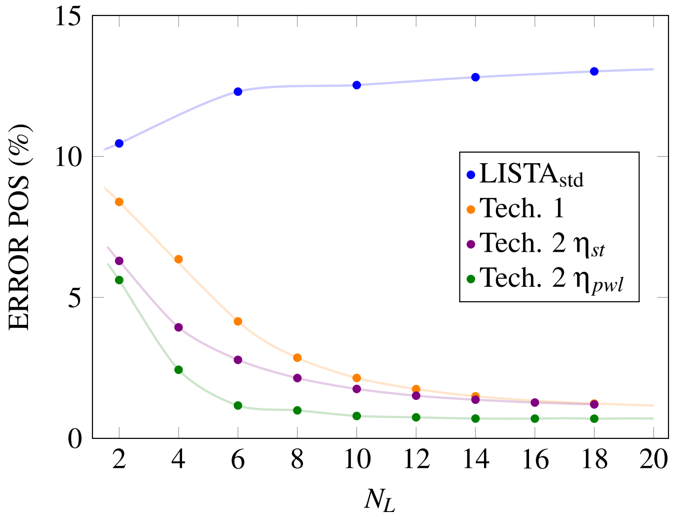


Figure 5.15 – Percentage of taps that had an improper attribution with the proposed neural network (Tech. 2).

Fig. 5.15 presents the percentage of taps which had an improper attribution [226]. It can be noticed that the proposed neural network (curves Tech. 2 $\eta_{st}$ and Tech. 2 $\eta_{pwl}$) better estimates the non-zero tap positions. This occurs because it was trained focused on finding the non-zero tap positions instead of estimate the tap values (which is the case of LISTA).

Fig. 5.16 presents the obtained NMSE as a function of the number of layers $N_L$ [226]. The performance obtained with the proposed neural network is also compared to LISTA, OLS and the results achieved in Section 5.3.2 with $\alpha = 0.02$ (curve Tech. 1).

From Fig. 5.16, it can be seen that the proposed neural network (curves Tech. 2 $\eta_{st}$ and Tech. 2 $\eta_{pwl}$) improves NMSE values compared to LISTA (curve LISTA$_{std}$). Indeed, using $\eta_r = \eta_{pwl}$ (curve Tech. 2 $\eta_{pwl}$) the NMSE value can decrease up to 10.8 dB compared to LISTA$_{\text{std}}$ and the results achieved are close to OLS. On the other hand, when $\eta_r = \eta_{st}$, Tech. 2 achieves similar results in terms of NMSE when compared to the one achieved in Section 5.3.2 (curve Tech. 1). When $\eta_r = \eta_{pwl}$, more parameters have to be calculated in the training phase compared to $\eta_r = \eta_{st}$. However $\eta_r = \eta_{pwl}$ enables the proposed neural network be better adjusted to sparse estimation requiring few layers.

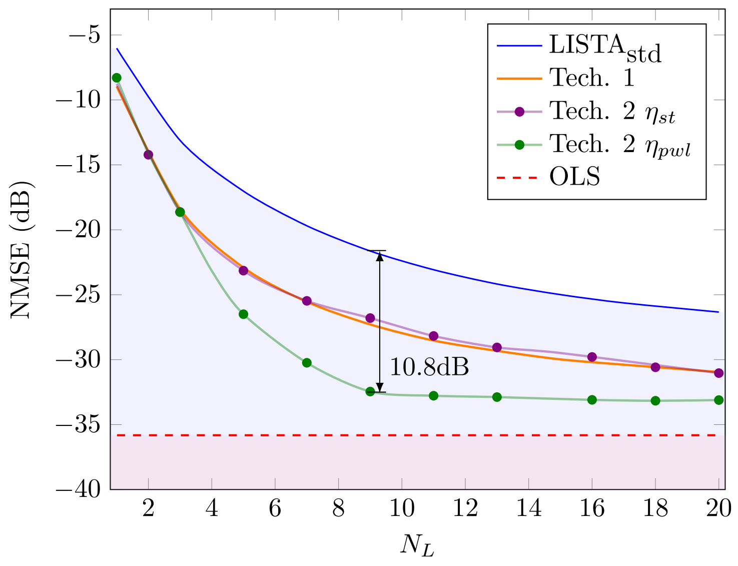


Figure 5.16 – NMSE × $N_L$ using "Tech. 1" and "Tech. 2".

In the simulations above, the channels in the sparse signal used in the training phase and the evaluation phase are generated with the same statistics. However, in real-world applications, mismatches may occur between the two phases. Therefore, it is essential for the trained models to be relatively robust to these mismatches.

In the next simulations, the impact of variation in statistics of the sparse signal used during training phase and evaluation phase is analyzed.

Fig. 5.17 shows the obtained NMSE when the signal to be estimated and the signals used during the training phase have different $\gamma$ values [226]. $N_L = 9$ and the training data has $\gamma = 0.1$. As expected, when the signal to be estimated is less sparse ($\gamma > 0.1$) than the signals used in the training data ($\gamma = 0.1$), the NMSE values achieved by "Tech. 1" and "Tech. 2" dissociate themselves from the NMSE of the OLS. However, it should be highlighted that "Tech. 2 $\eta_{pwl}$" leads to better NMSE values suggesting that it has some degree of robustness when sparsity mismatch occurs.

## 5.5 Shrinkage Functions in LISTA

This section explores alternative shrinkage functions to be used in LISTA. The objective is to analyze their impact in terms of estimation quality.

The traditional shrinkage function $\eta_{st}$ used in LISTA is defined by (3.16) and it is shown in Fig. 5.18.a. Recently, cubic B-splines shrinkage function has been used in LISTA with several parameters spread uniformly over the dynamic range of the signal to improve the estimation qual-

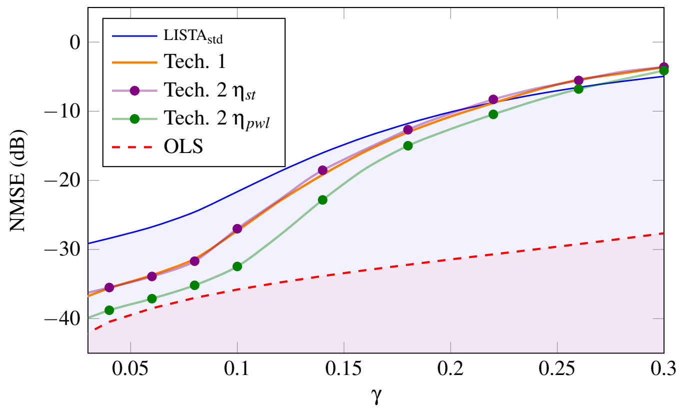


Figure 5.17 – NMSE $\times$ $\gamma$ using “Tech. 1” and “Tech. 2”.

ity [200]. As an alternative to these solutions, three shrinkage functions controlled by a small number of learnable parameters that can vary across the layers of the LISTA network are considered here:

**Exponential ($\eta_{exp}$):** Let $\theta = \{\theta_1, \theta_2, \theta_3\}$, the exponential shrinkage function is defined by [227]:

$$\eta_{exp}(x;\theta) = \theta_2 x + \theta_3 x exp\left(-\frac{x^2}{2\theta_1^2}\right) \tag{5.29}$$

**Piecewise Linear ($\eta_{pwl}$):** Let $\theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$, this function has five segments [227]:

$$\eta_{pwl}(x;\theta) = \begin{cases} \theta_3 x, & if \quad |x| \leq \theta_1 \\ sgn(x)[\theta_4(|x|-\theta_1)+\theta_3\theta_1], & if \quad \theta_1 < |x| \leq \theta_2 \\ sgn(x)[\theta_5(|x|-\theta_2)+\theta_4(\theta_2-\theta_1)+\theta_3\theta_1], & if \quad \theta_2 < |x| \end{cases} \tag{5.30}$$

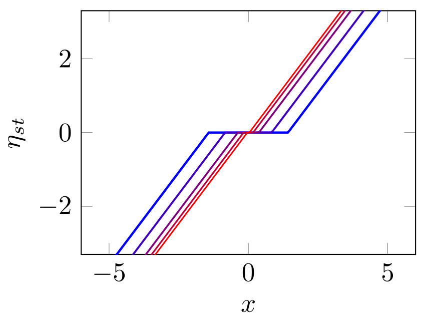
**Spline ($\eta_{spl}$):** The spline shrinkage function is defined by:

$$\eta_{spl}(x;\theta) = \theta_2 x + \theta_3 x \beta\left(\frac{x}{\theta_1}\right) \tag{5.31}$$

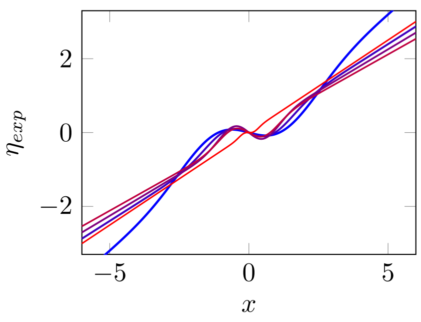
where $\theta = \{\theta_1, \theta_2, \theta_3\}$ and $\beta$ is the cubic B-spline [228]:

$$\beta(z) = \begin{cases} \frac{2}{3} - |z|^2 + \frac{|z|^3}{2}, & if \quad 0 \leq |z| \leq 1 \\ \frac{1}{6}(2-|z|)^3, & if \quad 1 \leq |z| \leq 2 \\ 0, & if \quad 2 \leq |z| \end{cases} \tag{5.32}$$

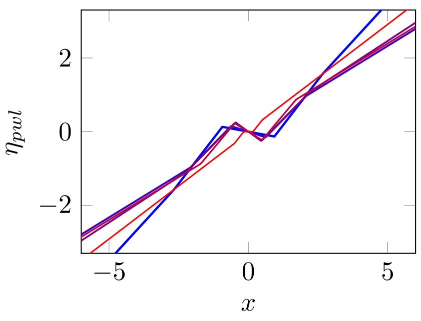
In Fig. 5.18, each color represents a different set of parameters used in each shrinkage function.



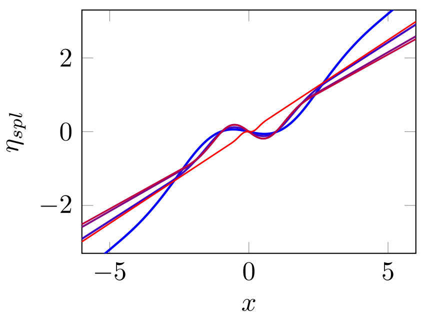
(a) Soft Threshold.



(b) Exponential.



(c) Piecewise Linear.



(d) Spline.

Figure 5.18 – Shrinkage functions for different parameters.

### 5.5.1 Application Case

The normalized mean squared error (NMSE) described by (5.2) is used to evaluate ISTA and LISTA estimation performance in terms of the number of iterations ($N_{it}$) and number of layers ($N_l$), respectively. $N_s$ is the number of realizations. In this work, $N_s = 10000$.

The system model is defined by (2.8), considering:

- $N = 120$.
- $M = 60$.
- $\mathbf{A}$ is i.i.d. Gaussian, with elements distributed $\mathcal{N}(0, M^{-1})$.
- $\mathbf{h}$ is Bernoulli-Gaussian, that is, its elements are i.i.d $\mathcal{N}(0,1)$ with probability $\gamma = 0.1$ and the others are set to 0.
- SNR = 30 dB.

First, the parameters dependence in ISTA performance is verified. Next, the LISTA performance is analyzed for the shrinkage functions presented in Section 5.5. The results are compared to the theoretical performance bound OLS.

### 5.5.2 Estimation Performance

#### 5.5.2.1 ISTA Performance

Fig. 5.19 presents the ISTA estimation performance for different values of the parameters β and λ [229].

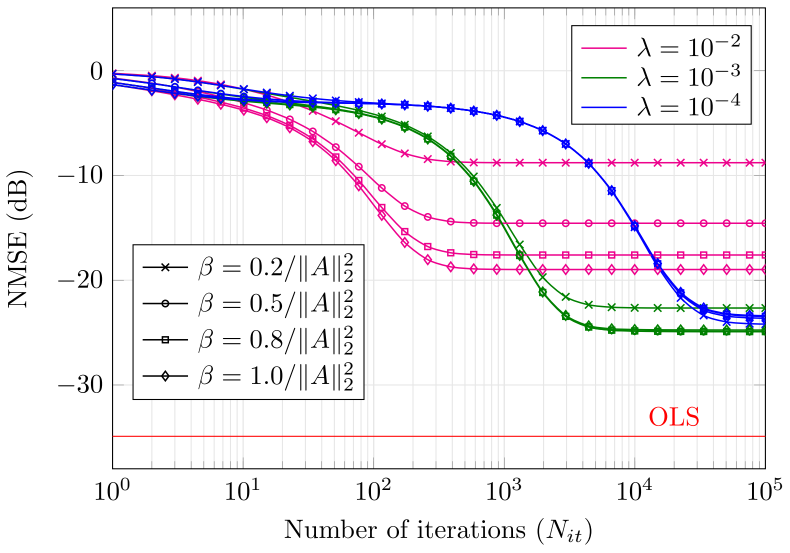


Figure 5.19 – ISTA estimation performance for different β and λ values.

This figure leads to the following observations [229]:

- Many iterations are required for ISTA convergence.
- In all considered cases, ISTA does not converge to a NMSE value close to the one achieved by OLS.
- ISTA converges to different NMSE values at different number of iterations depending on the β and λ values.

These results highlight the importance of the parameters choice as they highly influence the ISTA performance in sparse signal estimation. Moreover, they point out that even with a lot of iterations, ISTA does not give an estimation result close to the one achieved by the theoretical performance bound OLS.

#### 5.5.2.2 LISTA Performance

In order to analyze the LISTA estimation performance, training data set $\{(\mathbf{y}^d, \mathbf{h}^d)\}_{d=1}^{D}$ with $D = 10000$ was used. For the testing phase, another set with a size of 10000 was used. In both phases, sets were generated independently with the same distribution parameters.

During the training phase of LISTA, the network adapts the parameters minimizing the loss function (4.3) through the Adam optimizer. Therefore, it is not necessary to choose the ISTA parameters $\beta$ and $\lambda$, LISTA learns its parameters during the training phase.

Fig. 5.20 presents the LISTA estimation performance in terms of NMSE value considering the traditional shrinkage function (3.16) and the functions described in Section 5.5 for different numbers of layers [229]. The parameters of the shrinkage functions were learned directly from independent realizations of the training data during the training phase.

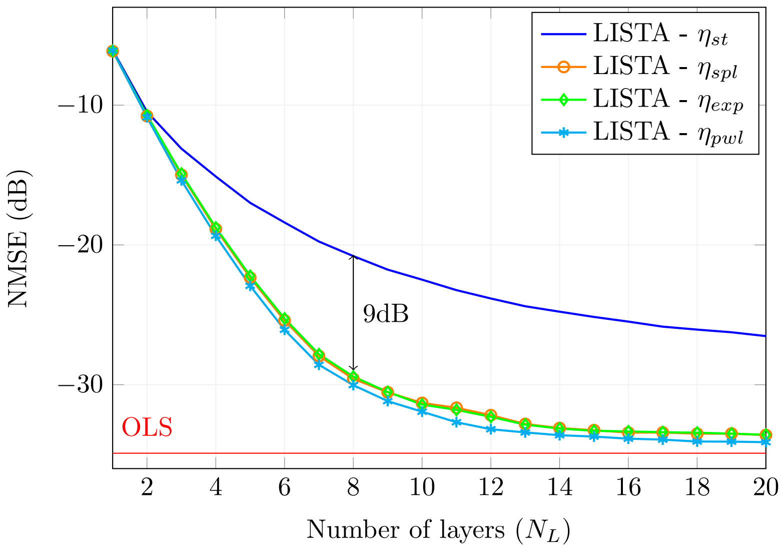


Figure 5.20 – LISTA estimation performance according to the number of layers.

From this figure, it can be seen that [229]:

- The NMSE values achieved by LISTA with the piecewise linear, exponential, and spline shrinkage functions are better than those achieved by LISTA with the traditional shrinkage function $\eta_{st}$.
- The use of piecewise linear, exponential, and spline shrinkage functions leads to NMSE value decrease up to 9 dB compared to the traditional shrinkage function.
- The use of one of the shrinkage functions presented in Section 5.5 allows LISTA to achieve

performance close to OLS in a reasonable number of layers.

- While the traditional shrinkage function (3.16) has only one parameter that is adjusted for each layer, the functions of Section 5.5 have three or five parameters. The results suggest that increasing the degree of freedom of the shrinkage function, better results in terms of NMSE can be achieved.

The computational complexity of one layer of LISTA and one iteration of ISTA are essentially the same [31]. Comparing ISTA and LISTA estimation performances by the results presented in Figs. 5.19 and 5.20, respectively, it can be noticed that [229]:

- Applying LISTA with only 6 layers, a NMSE equals to $-24$ dB can be reached. On the other hand, more than 3600 iterations of ISTA would be required to achieve this NMSE value.
- LISTA network generates estimates with lower NMSE value and less computational complexity (significantly fewer matrix-vector multiplications) than ISTA.
- With LISTA, the parameters are internally optimized during the training phase.

Therefore, the obtained results show that LISTA generates better estimate than ISTA. Furthermore, combining LISTA with one of the shrinkage function presented in Section 5.5, an estimate close to the OLS can be achieved using few layers of LISTA.

The above simulations consider that the training data and the signal of interest have similar characteristics. However, the signal to be estimated can have different characteristics compared to those of the training data used to training the network. In order to analyze the robustness of the network generated by LISTA, two different cases were simulated:

- Case 1: the signal to be estimated and the signals used during the training phase have different SNR values.

  Fig. 5.21 shows LISTA behaviour varying the SNR value of the signal to be estimated. The network has $N_L = 20$ and the training data has $SNR = 30$ dB [229].

  It can be noticed that the performance decreases, as outside the zone near to $SNR = 30$ dB, the obtained NMSE values are not close to those produced by OLS. That being said, even if for $SNR \leq 20$ dB, all shrinkage functions lead to similar performances, the functions $\eta_{spl}$, $\eta_{exp}$, and $\eta_{pwl}$ achieve better results than the traditional function for $SNR \geq 20$ dB.

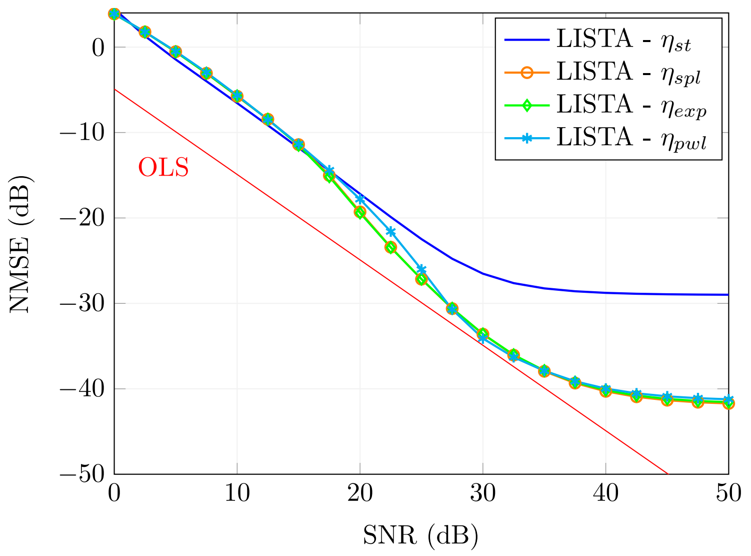


Figure 5.21 – LISTA estimation performance varying the SNR of the signal to be estimated.

These results suggest that the functions presented in Section 5.5 can be an interesting solution to estimate signals with high SNR. Better performance to estimate signals with a wider range of SNR can be achieved using one of these functions than if the $\eta_{st}$ is used. It is very useful for applications where the SNR value of the signal to be estimated is variable, for example in wireless communication systems.

- Case 2: the signal to be estimated and the signals used during the training phase have different $\gamma$ values.

  Fig. 5.22 shows the LISTA estimation performance varying the $\gamma$ of the signal to be estimated. $N_L = 20$ and the training data has $\gamma = 0.1$ [229].

  As expected, when the signal to be estimated is less sparse ($\gamma > 0.1$) than the signals used in the training data ($\gamma = 0.1$), the NMSE values achieved by LISTA dissociate themselves from the NMSE of the OLS. However, it should be highlighted that the use of one of the shrinkage functions presented in Section 5.5 instead of the traditional shrinkage function (3.16) leads to better NMSE values.

  As well as SNR, the sparsity of the signal to be estimated may not be fixed. As can be seen in Fig. 5.22, LISTA with $\eta_{st}$ has some degree of robustness when the sparsity varies. However, this robustness clearly increase when one of the shrinkage functions proposed is used.

The results indicate that the choice of the shrinkage function is an important factor for the esti-

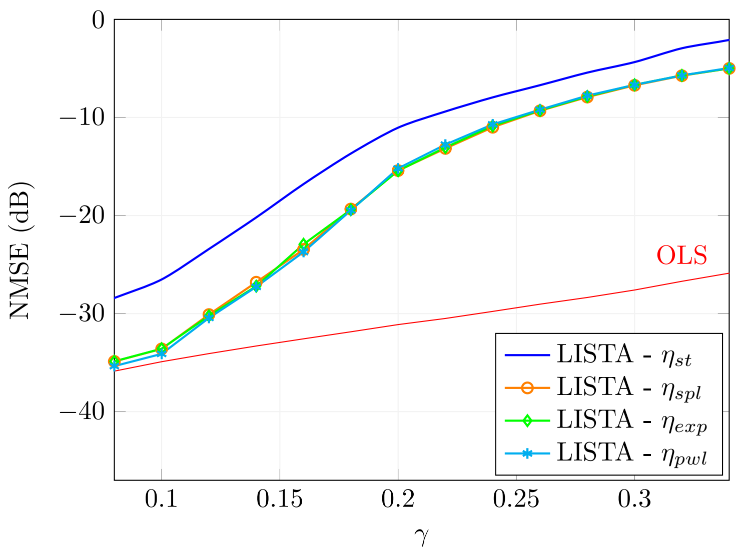


Figure 5.22 – LISTA estimation performance varying the $\gamma$ of the signal to be estimated.

mation performance of LISTA. Indeed, more flexible shrinkage functions, like exponential, spline, or piecewise linear led to better estimation performance than the traditional shrinkage function $\eta_{st}$. The piecewise linear function has a small advantage over the other shrinkage functions when the characteristics of the training set and the signal to be estimated are the same (Fig. 5.20). On the other hand, in terms of robustness to SNR mismatch, the functions splines and exponential present a slight advantage (Fig. 5.21).

Fig. 5.20 and Fig. 5.21 point out that using a training data with a determined value of SNR and $\gamma$ to estimate a signal with different values for these parameters leads to decreased LISTA estimation performance. To improve the LISTA performance, in addition to change the shrinkage function to one of the presented in Section 5.5, the training data could be modified.

## 5.6 Performance Comparison

This section presents a performance comparison between some sparse recovery algorithms reported in the literature and the approaches proposed in this thesis. The algorithms and neural network analyzed here are:

- OLS: theoretical performance bound Oracle Least Square;
- MP: Matching Pursuit (see Section 3.3.1);
- OMP: Orthogonal Matching Pursuit (see Section 3.3.2);

- MPLS: Matching Pursuit based on Least Squares (see Section 5.1);
- LISTA$_s$: Learned Iterative Shrinkage-Thresholding Algorithm with the traditional shrinkage function $\eta_{st}$ defined by (3.16) (see Section 4.2.1);
- LISTA$_{pwl}$: Learned Iterative Shrinkage-Thresholding Algorithm with the piecewise linear shrinkage function $\eta_{pwl}$ defined by (5.30) (see Section 5.5);
- Tech. 1: Proposition to improve the LISTA performance (see Section 5.3);
- PNN$_s$: Proposed Neural Network (Tech. 2) with the shrinkage function $\eta_{st}$ defined by (3.16) (see Section 5.4);
- PNN$_{pwl}$: Proposed Neural Network (Tech. 2) with the piecewise linear shrinkage function $\eta_{pwl}$ defined by (5.30) (see Section 5.4).

They are evaluated in terms of NMSE described by (5.2), the percentage of non-zero taps positions correctly found ($P_c$) and the percentage of taps which had an improper attribution ($P_e$). The number of multiplications ($N_M$) is used as the parameter of the algorithm computational complexity.

Firstly the floating-point arithmetic implementation is considered (see Section 5.6.1). Then the estimation performances are analyzed using fixed-point arithmetic (see Section 5.6.2).

The system model is defined by (2.8), considering:

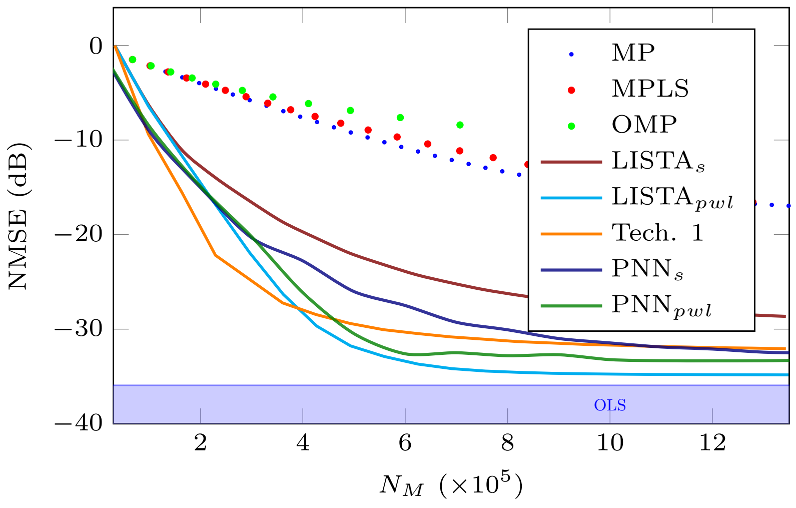
- $N = 256$.
- $M = 128$.
- **A** is i.i.d. Gaussian, with elements distributed $\mathcal{N}(0, M^{-1})$.
- **h** is Bernoulli-Gaussian, that is, its elements are i.i.d $\mathcal{N}(0,1)$ with probability $\gamma = 0.1$ and the others are set to 0.
- SNR = 30 dB.
- $N_s = 10000$.
- The training data sets used in the neural networks analyzed have $D = 10000$ pairs of elements.

- The testing phase of the neural networks analyzed have another sets with a size of 10000 pairs of elements.
- The Adam optimizer was used in the neural networks analyzed.
- $\alpha = 0.02$ for Tech.1.
- $\tau = 0.2$ for PNN$_s$ and PNN$_{pwl}$.

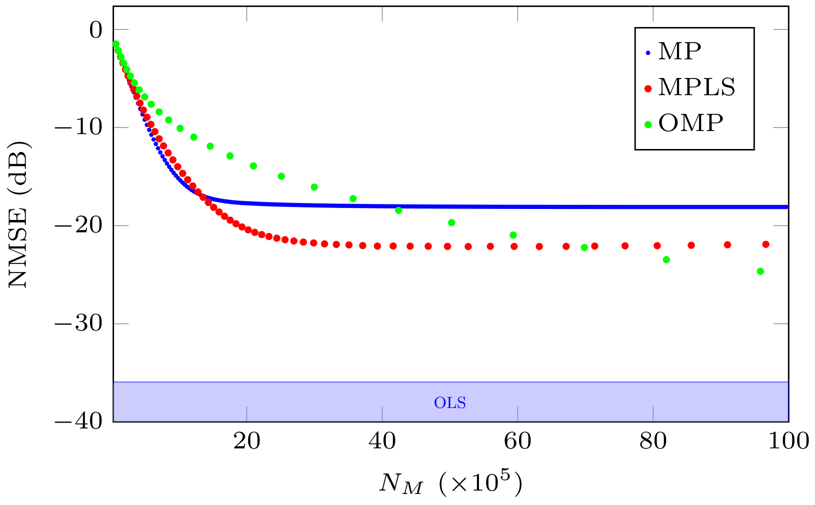
### 5.6.1 Estimation Performance: Floating-point Arithmetic

Fig. 5.23 presents the estimation performances for floating-point arithmetic in terms of number of multiplications ($N_M$). The results indicate that using neural network for estimating the sparse signal leads to better estimation performance (lower NMSE value) than using one of the greedy algorithm analyzed (MP, MPLS, and OMP). Indeed, the NMSE values achieved by the greedy algorithms are very far from the value reached by the OLS. Moreover, it can be seen in Fig. 5.23 that LISTA$_{pwl}$, Tech. 1, PNN$_s$ and PNN$_{pwl}$ improve the estimation performance compared to LISTA$_s$. In addition, LISTA$_{pwl}$ and PNN$_{pwl}$ are the ones which achieve results closer to the one of the theoretical performance bound OLS. As $\eta_{pwl}$ has more parameters than $\eta_s$, this has an important influence on the estimation performance of LISTA and PNN as can be noticed when comparing the results of LISTA$_s$/LISTA$_{pwl}$ and PNN$_s$/PNN$_{pwl}$.

Fig. 5.24 shows the percentage of non-zero taps positions correctly found. It can be noticed that the greed algorithms require more $N_M$ to correctly found more than 80% of the non-zero tap positions. In contrast, LISTA$_s$ and LISTA$_{pwl}$ correctly found more than 80% of the non-zero tap positions faster than the others techniques, that is, with fewer $N_M$. However, one problem of LISTA is that it attributes non-zero values to a lot of taps, so even if it finds a lot of non-zero tap positions, it also attributes small values to wrong taps. This can be noticed in Fig. 5.25 that provides the percentage of taps with an improper attribution. This characteristic of LISTA is more accentuated in LISTA$_{pwl}$ that assigns non-zero values to almost all taps as can be noticed in Fig. 5.24 and Fig. 5.25. Even if it has a big $P_e$, it doesn't influence a lot the NMSE because the values attributed to the taps are close to zero. However, LISTA$_{pwl}$ is not appropriated to be use if the knowledge of the non-zero tap positions is important. Fig. 5.25 shows that Tech. 1, PNN$_s$ and PNN$_{pwl}$ have lower $P_e$ than LISTA$_s$ and LISTA$_{pwl}$. Indeed, $P_e < 2\%$ can be achieved with $N_M = 3 \times 10^5$, $N_M = 5 \times 10^5$, and $N_M = 13 \times 10^5$ using PNN$_{pwl}$, PNN$_s$, and Tech. 1, respectively. These results suggest that PNN$_{pwl}$ is the best technique to found the non-zero tap positions. This



(a)



(b)

Figure 5.23 – Estimation performance for floating-point arithmetic in terms of number of multiplications ($N_M$).

occurs because PNN focuses on estimating the non-zero tap positions instead of estimate their values (see Section 5.4).

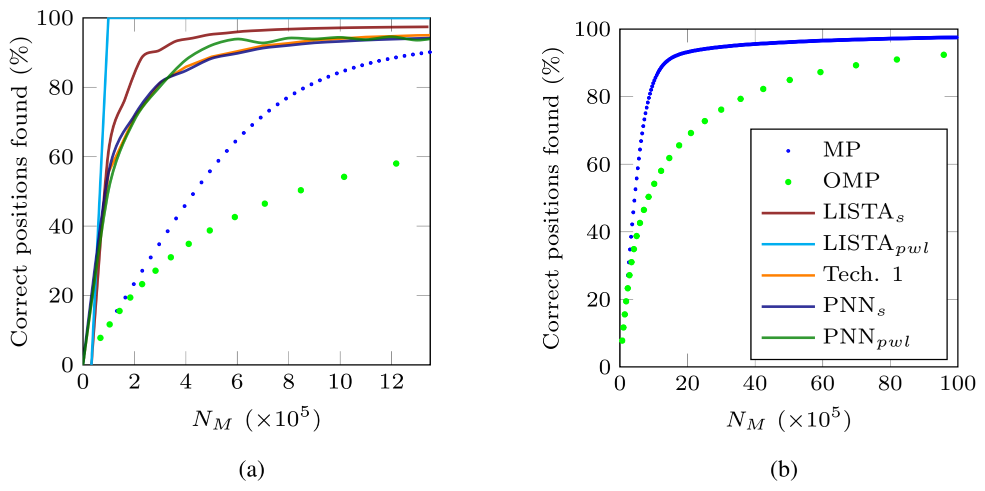


Figure 5.24 – Percentage of non-zero taps positions correctly found ($P_c$).

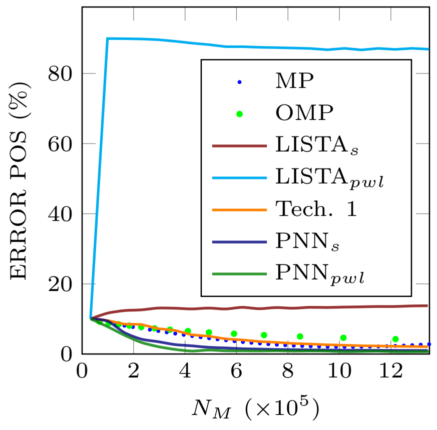


Figure 5.25 – Percentage of taps which had an improper attribution ($P_e$).

These results indicate PNN$_{pwl}$ as the best technique to estimate a sparse signal. It has the fewest percentage of taps which had an improper attribution and it achieves NMSE value close to the ones of the OLS. Even LISTA$_{pwl}$ leads to NMSE values a little lower than the ones of PNN$_{pwl}$, LISTA$_{pwl}$ has the disadvantage of assigning non-zero values to a lot of taps which should have a zero value, resulting in a big $P_e$.

### 5.6.2 Estimation Performance: Fixed-point Arithmetic

Sparse signal recovery on hardware is challenging since it consists of dot product operations and complex matrix inversion (depending on the sparse recovery algorithm chosen). A hardware implementation has to take into account its cost and its performance. In general, more hardware multipliers and more memory imply more power consumption. However, some applications have hardware constraints requiring low-power and low-cost signal processing operations targeting longer battery life or more compact physical devices.

One alternative is to reduce the number of bits used in the hardware implementation to represent the values that will be calculated. Processors designed for extremely low power consumption or very small size usually only provide multiplication instructions that are limited with respect to the bit size of the words [230]. However, this reduction often leads to a degradation of the algorithm performance. Therefore, this negative impact should be analyzed.

Floating-point arithmetic is complex and requires more area than fixed-point arithmetic increasing hardware complexity [231]. Moreover, fixed-point implementation reduces the amount of switching activity on wires and logic gates, leading to the reduction of power dissipation [232]. Taking these into account, the algorithms evaluated here were implemented using fixed-point arithmetic. In this thesis, $Qi.f$ is used to represent the quantization format where $i$ is the amount of integer bits and $f$ is the amount of fractional bits.

The residual vector $\mathbf{b}_i$ of the OMP algorithm is always orthogonal to the columns that have already been selected. Therefore, theoretically, there will be no columns selected twice and the set of selected columns is increased through the iterations. However, when the OMP is implemented with limited data representation, there is an imperfect orthogonalization caused by inaccurate least square calculation. For this reason, a choice should be made when implementing it:

- OMP$_n$: the implementation can be forced to always choose a new non-zero tap, that is, different from the ones selected in previous iterations keeping the original sense of orthogonality of the OMP.

- OMP$_r$: the implementation allows the non-zero tap to be re-selected in others iterations considering that orthogonality is not really achieved.

Fig. 5.26 shows the estimation performance in terms of NMSE varying the number of multiplication operation ($N_M$) considering the two alternatives of OMP implementation for $Q8.8$. It can be seen that when the non-zero taps can not be selected twice (OMP$_n$) the NMSE value is bigger than the OMP$_r$ case. It occurs because OMP$_n$ starts selecting wrong non-zero taps faster than OMP$_r$, that is, with less iterations. Therefore, increasing the error estimation.

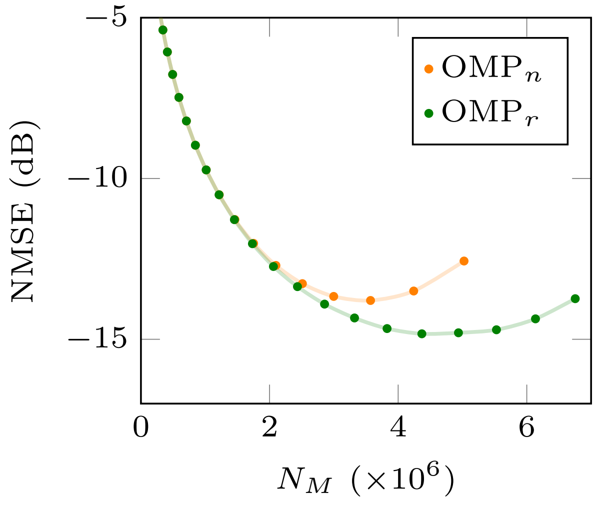


Figure 5.26 – $\mathrm{OMP}_n$ and $\mathrm{OMP}_r$ estimation performances for $Q8.8$ (16 bits).

Fig. 5.27 and Fig. 5.28 present the estimation performances of the MP, OMP and MPLS algorithms considering 16-bits implementation ($Q8.8$) and 24-bits implementation ($Q8.16$), respectively. Each point in the curves represents one iteration of the algorithm. It can be noticed that for the same quantity of multiplication operation ($N_M$) the OMP algorithm did less iteration (there are less points in the green curves than in the other curves). This occurs because each iteration of the OMP algorithm solves the least square problem which consumes a lot of multiplication operations. Indeed, as iteration order goes up, the OMP algorithm computational complexity of the iteration increases due to the expansion of the matrix which has to pass by the pseudo-inversion process. On the other hand, the MP algorithm has the simplest iteration calculations leading to more iterations (many blue points in the figures) for a same $N_M$ value.

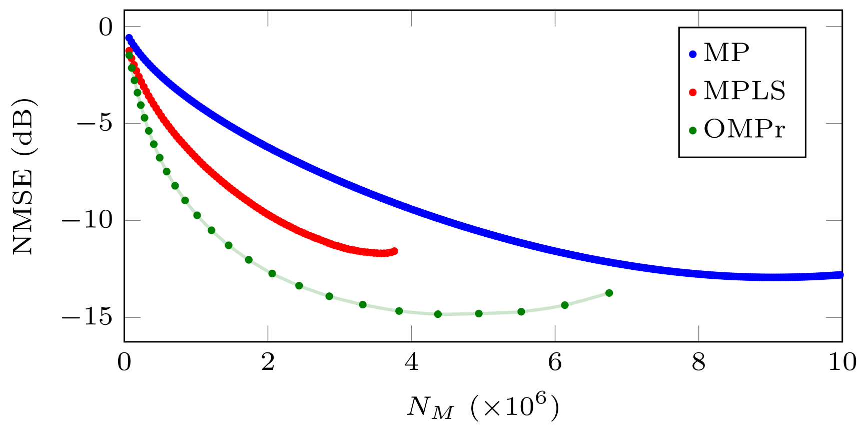


Figure 5.27 – MP, OMP and MPLS estimation performances for $Q8.8$ (16 bits).

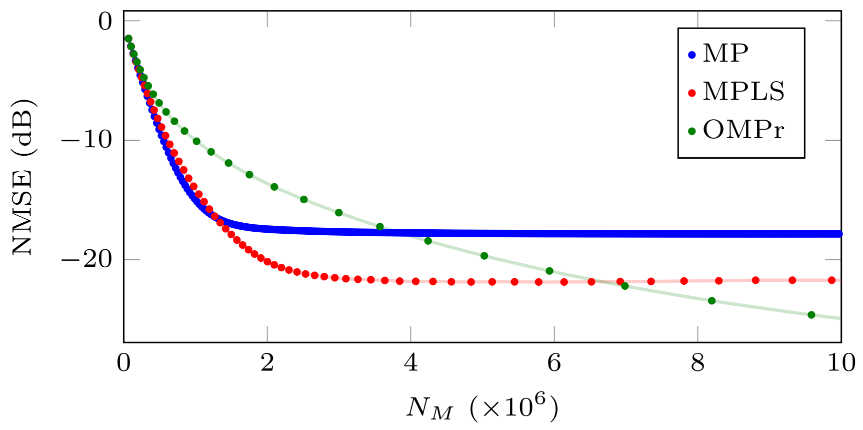


Figure 5.28 – MP, OMP and MPLS estimation performances for $Q8.16$ (24 bits).

The MP, OMP and MPLS algorithms start considering that all taps have zero values. Throughout each iteration they increase or maintain the amount of non-zero taps. This can be seen in Fig. 5.27 and Fig. 5.28. Initially the value of the NMSE decreases with the increase of $N_M$. This is because the algorithms find the positions of the taps that have non-zero values and assign values to these taps. However, as the number of iterations increases, the algorithms start to assign non-zero values to taps that should have zero values, so the NMSE increases, i.e., the estimation error increases. For $Q8.8$ (Fig. 5.27) the algorithms were interrupted after increasing the NMSE values as the number of iterations increased.

Considering the 16-bit implementation ($Q8.8$ - Fig. 5.27), the OMP algorithm presents the best performance in terms of achieving lower NMSE for the same value of $N_M$ when compared to the MP and the MPLS algorithms for $N_M < 6.5 \times 10^6$. On the other hand, for the 24-bit implementation ($Q8.16$ - Fig. 5.28), the MPLS algorithm has the best performance (lowest NMSE value) for $N_M < 6.5 \times 10^6$. If more multiplication operations are allowed then the OMP achieves better performance (lowest NMSE). Indeed, Fig. 5.29 shows the minimum NMSE value achieved by the MP, OMP and MPLS algorithms for 24-bits implementation. As said before, each MPLS and MP iteration consumes fewer multiplication operations than one OMP iteration. Thus, it is possible to find the non-zero tap positions faster with the MP and MPLS algorithms and thus estimate the sparse signal faster than the OMP algorithm, leading to a lower NMSE value for a lower $N_M$.

Therefore, it can be seen from Fig. 5.27 and Fig. 5.28 that depending on the amount of bits, the algorithm that has the best estimation performance for a small $N_M$ value varies. In other words, it would be better to choose the OMP algorithm to sparse signal estimation if 16-bits implementation is used. On the other hand, if 24-bits implementation is used, it would be better to choose the

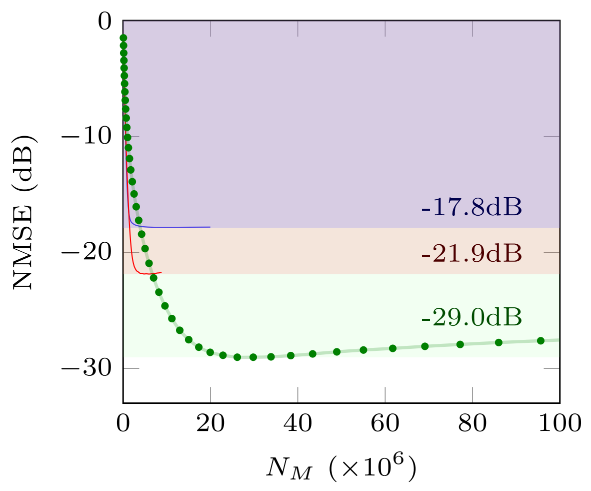
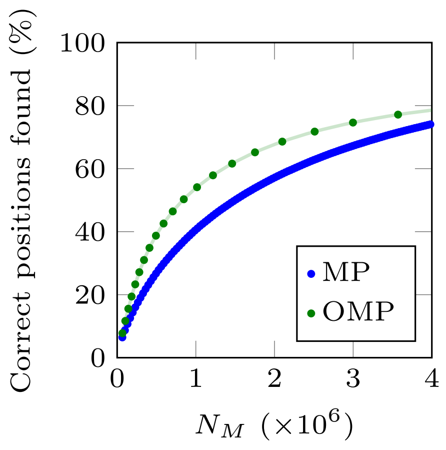


Figure 5.29 – Minimum NMSE value for 24 bits ($Q8.16$).

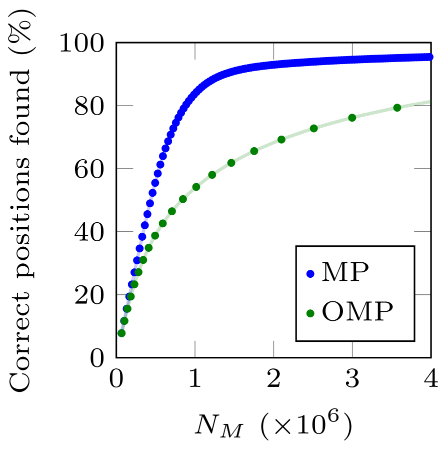
MPLS algorithm so it would reduce the computational complexity ($N_M < 6.5 \times\ 10^6$) and achieve a lower NMSE value. This difference in the estimation performance of 16-bits and 24-bits implementations occurs because with more bit accuracy (more bits), the fact that the MPLS algorithm finds the correct positions of the non-zero taps with fewer iterations than the OMP algorithm is a big advantage for the estimated signal calculation. In addition, as expected, increasing the number of bits improves the estimation performance of the algorithms, i.e., the achieved NMSE value decreases.

Fig. 5.30 shows the percentage of non-zero taps positions correctly found according to the number of multiplication operations performed. The MPLS algorithm is not represented here because it finds the positions in the same way as the MP algorithm, the difference between these algorithms lies in the calculation of the non-zero tap values and not in the way to determine their positions. It can be seen in this figure that with 16-bits implementation, for the same value of $N_M$, the OMP finds more correct positions, which contributes to the result presented in Fig. 5.27. On the other hand, with 24-bits implementation (see Fig. 5.30.b), the MP finds much more correct non-zero tap positions, quickly reaching more than 80% of non-zero tap positions. These positions are the same as those found by the MPLS algorithm. This result reinforces the good performance achieved by the MP and MPLS algorithms when compared to the OMP in Fig. 5.28.

Fig. 5.31 presents the minimum NMSE value achieved for each algorithm considering 8 fractional bits ($Q8.8$) and 16 fractional bits ($Q8.16$) implementations. It is worth highlight that these minimums are achieved considering different computational complexities, i. e., different $N_M$ values for each algorithm (see Fig. 5.27 and Fig. 5.28). In addition, Fig. 5.31 shows in the rose line



(a) $Q8.8$



(b) $Q8.16$

Figure 5.30 – Percentage of non-zero taps positions correctly found using the MP and OMP algorithms.

the quantization noise imposed by bit limit precision. It is defined by (5.33) varying the number of fractional bits. The results are compared to the OLS (blue curve). It has the previous knowledge of the non-zero tap positions and calculates their non-zero coefficients with the floating-point least square solution. Therefore, these thresholds represent the optimal NMSE achievable limit (OLS) and the limit due to bit accuracy ($B_l$). It can be noticed that the MP, MPLS and OMP don't achieved the limit boundaries.

$$B_l = \frac{2^{-2f} \times N \times N_s}{\sum ||h||_2^2} \tag{5.33}$$

Sparse signal estimation using neural network has advantages when it is implemented in fixed-point arithmetic. The neural network training does not necessarily have to be done in hardware, so floating-point arithmetic can be used to train the network. Once trained, the sparse signal estimation is done using the neural network with fixed-point arithmetic.

Fig. 5.32 shows the estimation performance, in terms of NMSE, for LISTA$_s$, LISTA$_{pwl}$, Tech. 1, PNN$_s$, and PNN$_{pwl}$ considering $Q8.8$, $Q8.16$ and floating-point arithmetic (represented as $f$, for example LISTA$_{sf}$ means the float implementation of LISTA$_s$) implementations. The results presented in Fig. 5.23 are showed in Fig. 5.32 to facilitate the comparison. The $Q8.16$ implementation well represents the floating-point arithmetic suggesting that using 16 bits for the fractional part is adequate. On the other hand, using 8 bits in the fractional part ($Q8.8$) decreases the estimation performance in all techniques. This decrease is bigger using Tech. 1, PNN$_s$ or PNN$_{pwl}$ than using

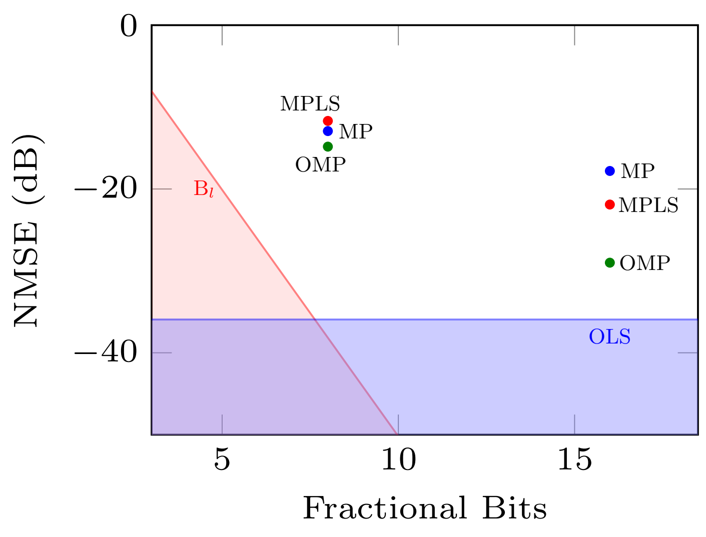


Figure 5.31 – Minimum NMSE value achieved.

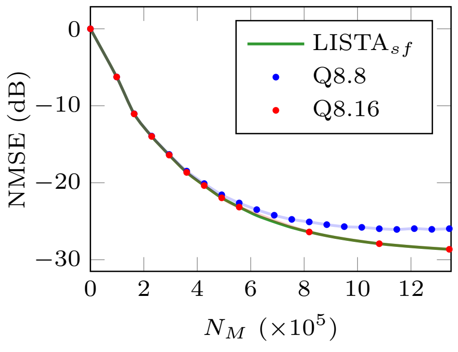
LISTA$_s$ or LISTA$_{pwl}$. This occurs because Tech. 1, PNN$_s$ and PNN$_{pwl}$ applied the least square solution to calculate the non-zero tap values. The complexity of the least square calculation suggests that the limited precision (to use 8 bits instead of 16 bit to represent the fractional part) has a greater influence on NMSE value than when the tap values are calculated directly by the network (it is the case of LISTA$_s$ and LISTA$_{pwl}$).

In addition, it can be noticed that LISTA$_{pwl}$ achieves lower NMSE value than the previous algorithms (see Fig. 5.27 and Fig. 5.28). Moreover, sparse signal estimation using LISTA$_{pwl}$ requires fewer multiplication operations ($N_M$) than the greedy algorithms MP, OMP and MPLS to achieve the same NMSE value. This occurs because during the LISTA execution there are not matrix inversion calculations leading to advantages over common sparse recovery algorithms. The LISTA$_{pwl}$ estimation performance does not vary much with the quantity of fractional bits. This represents an advantage to use the $Q8.8$ implementation which consumes less power and memory because it uses fewer bits.

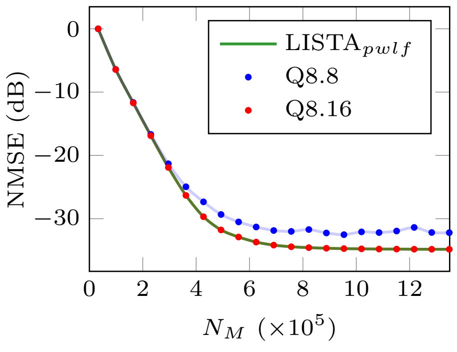
Fig. 5.33 presents the percentage of non-zero taps positions correctly found ($P_c$) using LISTA$_s$, LISTA$_{pwl}$, Tech. 1, PNN$_s$, and PNN$_{pwl}$ considering $Q8.8$, $Q8.16$ and floating-point arithmetic implementations. The results presented in Fig. 5.24 are showed in Fig. 5.33 to facilitate the comparison. Tech. 1 provides the biggest difference when the implementation is changed from $Q8.8$ to $Q8.16$. For the other techniques, using $Q8.8$, the $P_c$ value doesn't decrease a lot. LISTA$_s$, LISTA$_{pwl}$, PNN$_s$, and PNN$_{pwl}$ achieve more than 80% of non-zero tap positions correctly found with $N_M < 4 \times 10^5$ while the MP algorithm achieves it only with $N_M = 8.88 \times 10^5$ (see Fig. 5.30b).

Fig. 5.34 gives the percentage of taps which had an improper attribution ($P_e$) using LISTA$_s$, Tech. 1, PNN$_s$, and PNN$_{pwl}$ considering $Q8.8$, $Q8.16$ and floating-point arithmetic implementations. The results presented in Fig. 5.25 are showed in Fig. 5.34 to facilitate the comparison. The results of LISTA$_{pwl}$ are not presented here because more than 80% of taps had an improper attribution as can be seen in Fig. 5.25. It can be noticed that PNN$_{pwl}$ leads to the lowest $P_e$. This occurs because PNN is focused on finding the non-zero tap positions, so the neural network is trained for that, resulting in better performance than if LISTA is used.

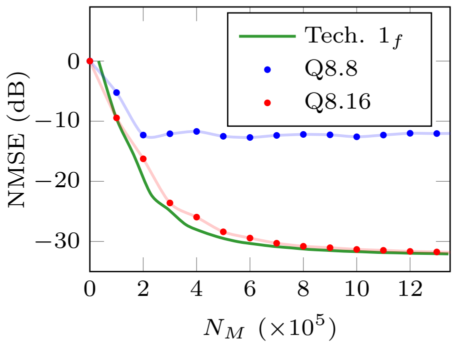
These results show that the performance of the techniques varies when the number of bit used in the fractional part changes. From these simulation, LISTA$_s$ is the technique that achieves better results for $Q8.8$ while for $Q8.16$ the PNN$_{pwl}$ leads to lowest percentage of taps with improper attribution and to good NMSE values.



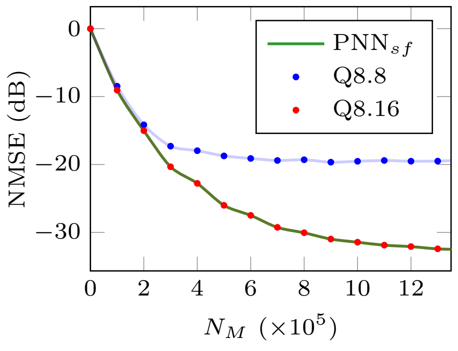
(a) LISTA$_s$



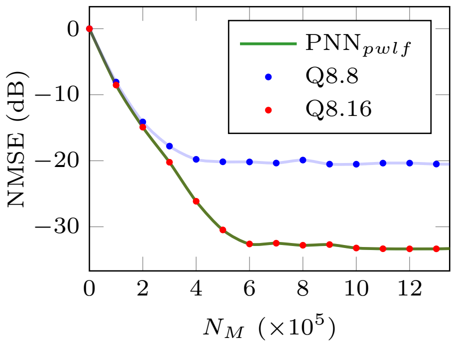
(b) LISTA$_{pwl}$



(c) Tech. 1

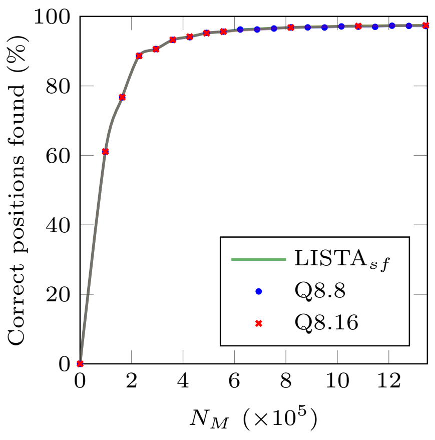


(d) PNN$_s$

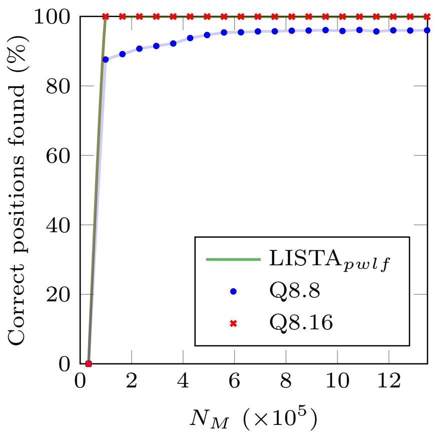


(e) PNN$_{pwl}$

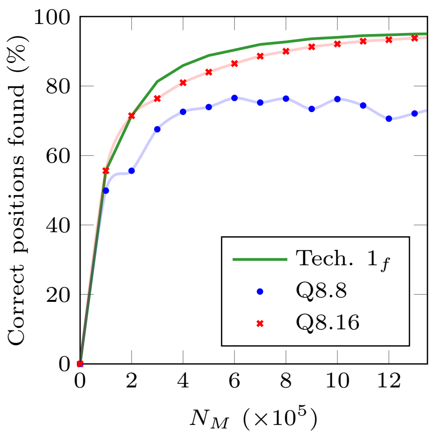
Figure 5.32 – Estimation Performance for $Q8.8$, $Q8.16$ and floating-point arithmetic using: (a) LISTA$_s$ (b) LISTA$_{pwl}$ (c) Tech. 1 (d) PNN$_s$ (e) PNN$_{pwl}$.



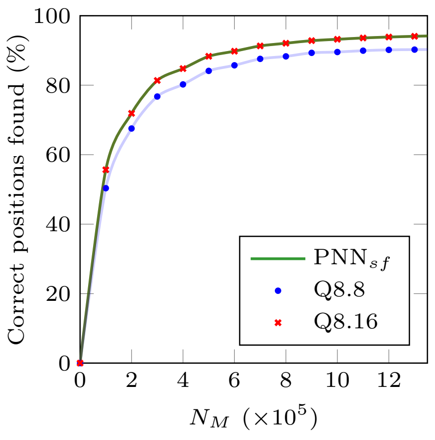
(a) LISTA$_s$



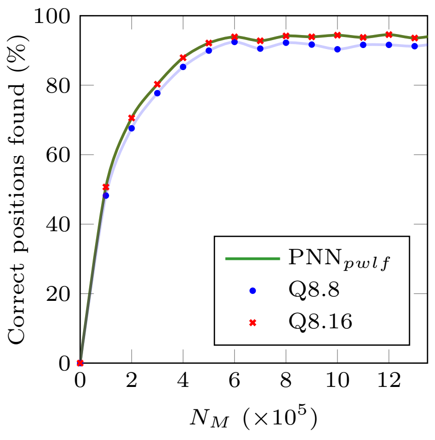
(b) LISTA$_{pwl}$



(c) Tech. 1

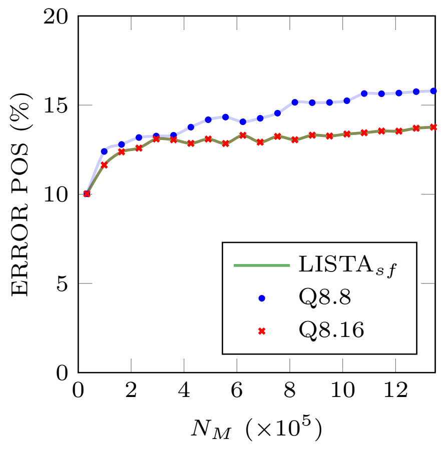


(d) PNN$_s$

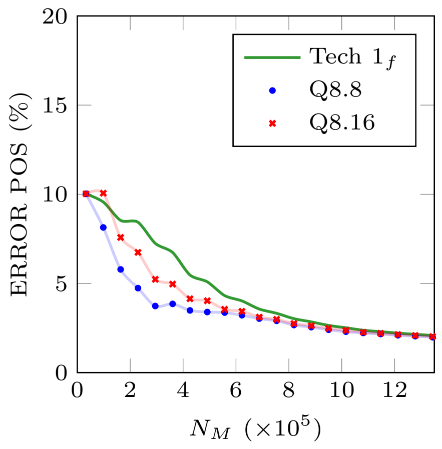


(e) PNN$_{pwl}$

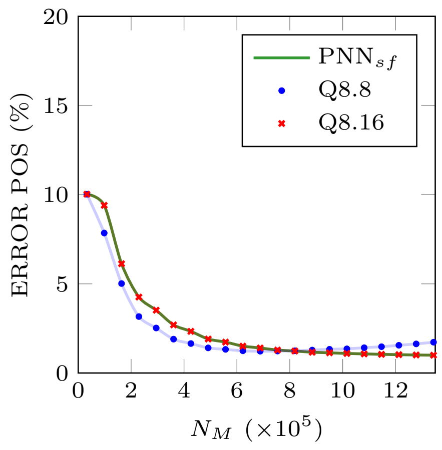
Figure 5.33 – Percentage of non-zero taps positions correctly found ($P_c$) for $Q8.8$, $Q8.16$ and floating-point arithmetic using: (a) LISTA$_s$ (b) LISTA$_{pwl}$ (c) Tech. 1 (d) PNN$_s$ (e) PNN$_{pwl}$.



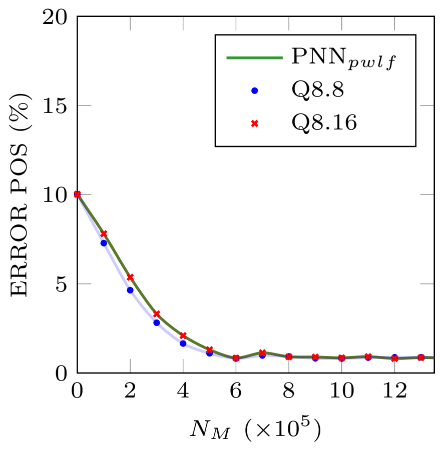
(a) LISTA$_s$



(b) Tech. 1



(c) PNN$_s$



(d) PNN$_{pwl}$

Figure 5.34 – Percentage of taps which had an improper attribution ($P_e$) for $Q8.8$, $Q8.16$ and floating-point arithmetic using: (a) LISTA$_s$ (b) Tech. 1 (c) PNN$_s$ (d) PNN$_{pwl}$.

# Chapter 6

# Conclusion and Perspectives

This chapter presents some conclusions and perspectives of this thesis work.

## 6.1 Conclusions

This thesis dealt with sparse channel estimation and exploration of new approaches to sparse signals estimation. High-quality channel estimation is very important to a reliable communication system. Several channels can be considered sparse. This characteristic can be used to improve their estimation. The compressive sensing and its sparse recovery algorithms are used in several areas and they can be applied to sparse channel estimation reducing the cost and the complexity of the estimation as well as the required amount of measurements. The work performed during this thesis mainly focused two aspects: sparse recovery algorithms and neural networks based compressive sensing.

Starting from the state of the art, several sparse recovery algorithms were studied and analyzed in Chapter 3. A performance analysis suggested that these algorithms better estimate sparse signals. In other words, the sparser the signal is, the smaller the estimation error will be. In addition, the algorithms BCS and OMP presented the lowest NMSE values (see Section 3.4). This study led to the first contribution of this thesis: the proposition of a greed algorithm called Matching Pursuit based on Least Squares (MPLS) which combines the Matching Pursuit and the Least Square algorithms (see Section 5.1). The results showed that the MPLS algorithm offers a better trade-off between estimation accuracy and computational cost than OMP algorithm, while producing similar results.

The study of these algorithms also led to the interest and study of the neural networks inspired by some of these algorithms, as discussed in Chapter 4. It resulted in other three contributions of this thesis presented in Sections 5.3, 5.4 and 5.5 to improve sparse signal recovery.

LISTA can be considered as one of the first neural network inspired by a sparse recovery algorithm. In this thesis, it is proposed to use the LISTA output to give a direction of the non-zero tap selection. The non-zero tap positions are chosen according to a threshold related to the biggest tap given by LISTA output. Then the signal estimate will be calculated by pseudo-inversion process (see Section 5.3). The obtained results show that using this approach, the NMSE value can be reduced up to 5.9 dB comparing to the LISTA estimation performance.

In addition, a new neural network focused on finding the non-zero tap positions was proposed (see Section 5.4). The obtained results show that the proposed network is a good approach to sparse signal recovery reducing up to 10.8 dB the NMSE value compared to LISTA and achieving performance close to OLS. Furthermore, fewer layers are required to achieve good estimates leading to less time required for training phase and less matrix-vector multiplications required during the evaluation phase.

Moreover, other shrinkage functions are suggested to replace the traditional shrinkage function of LISTA. Their estimation performances are analyzed in Section 5.5. The obtained results indicate that the estimates generated by the LISTA network require fewer matrix-vector multiplications than existing algorithms with optimally tuned regularization parameters. Furthermore, choosing an adequate shrinkage function, the LISTA can achieve NMSE values close to the theoretical performance bound with a few layers.

The results of this thesis reinforce the suitability of neural network in sparse signal estimation as an alternative to the sparse recovery algorithms. The approaches developed in this work can be extended and used to sparse signal estimation in several domains such as those presented in Section 2.3.2.

Most of the works covered in this thesis were published in appropriate conferences and journal, and those publications are listed in Appendix A.

## 6.2 Perspectives

As an extension of this thesis, the points below can further improve this work:

- Regarding the sparse recovery algorithms, a lot of them assume that the signal's sparsity is known. However, in several applications this is not true. Thus, it is necessary to develop sparse recovery algorithms that do not need this information and that are able to be adaptive to time changes. One other approach is to develop a sparsity estimator such as the proposed in [233].

- Although the use of neural network leads to performance improvements, it presents some challenges. The optimization algorithm, the loss function, and training strategies influence a lot the NN estimation performance. Therefore, they can be modified to try improve the sparse signal estimation performance.

- It should be interesting to explore the use of alternative functions $f(x)$ in the proposed neural network, so as to their estimation performance impact.

- Another opportunity for future work would be to make the function $\eta_r$ (see (5.24)) of the proposed neural network variable according to the layer. For example, to use a $\eta_{r1}$ function in the first layers and after to use a different $\eta_{r2}$ function.

- The step proposed in Fig. 5.10 can be applied in other neural networks. For instance, to replace the first step "LISTA" for another neural network for example LAMP and to use the estimate given by LAMP as $\hat{\mathbf{h}}$ in Fig. 5.10.

- The obtained results show that characteristics (e.g. SNR or $\gamma$) of the training set must match with those of the signal to be estimated, otherwise it can lead to some degradation in terms of NMSE. However, sometimes these characteristics vary, so a better choice of the training set should be done. For example, use with different SNR to better estimate a signal with variable SNR. Some simulations should be done to find the best way to compose the training set.

- As several channels are more complex in reality than their mathematical models and change over time, general NN that dynamically adapts to varying channel conditions is desirable. Therefore, efficient hardware implementations of the training phase are required for online training situations.

# Appendix A

# List of Publications

- MARQUES, E. C.; MACIEL, N.; NAVINER, L.; CAI, H.; YANG, J. “A Review of Sparse Recovery Algorithms”. In IEEE Access, vol. 7, pp. 1300-1322, 2018. doi: 10.1109/ACCESS.2018.2886471.

- MARQUES, E. C.; MACIEL, N.; NAVINER, L.; CAI, H.; YANG, J. “Nonlinear Functions in Learned Iterative Shrinkage-Thresholding Algorithm for Sparse Signal Recovery”. In IEEE International Workshop on Signal Processing Systems (SiPS 2019), Nanjing, China, 2019.

- MARQUES, E. C.; MACIEL, N.; NAVINER, L.; CAI, H.; YANG, J. “Deep Learning Approaches for Sparse Recovery in Compressive Sensing”. In International Symposium on Image and Signal Processing and Analysis (ISPA 2019), Dubrovnik, Croatia, 2019. doi: 10.1109/ISPA.2019.8868841.

- MARQUES, E. C.; MACIEL, N.; NAVINER, L.; CAI, H.; YANG, J. “Compressed Sensing for Wideband HF Channel Estimation”. In International Conference on Frontiers of Signal Processing (ICFSP 2018), Poitiers, France, pp. 1-5, 2018. doi: 10.1109/ ICFSP.2018.8552050.

- MACIEL, N.; MARQUES, E. C.; NAVINER, L. “Sparsity Analysis using a Mixed Approach with Greedy and LS Algorithms on Channel Estimation”. In International Conference on Frontiers of Signal Processing (ICFSP 2017), Paris, France, pp. 91-95, 2017. doi: 10.1109/ICFSP.2017.8097148.

Other publications:

- MACIEL, N.; MARQUES, E. C.; NAVINER, L.; CAI, H.; YANG, J. “Reliability Analysis of NAND-Like Spintronic Memory”. In Microeletronics Reliability Journal, 2019. doi: 10.1016/j.microrel.2019.06.024.

- MACIEL, N.; MARQUES, E. C.; NAVINER, L.; CAI, H.; YANG, J. “Voltage-Controlled Magnetic Anisotropy MeRAM Bit-Cell over Event Transient Effects”. In Journal of Low Power Electronics and Applications, vol. 9, 2019. doi: 10.3390/jlpea9020015.

- PAIVA, N. M.; MARQUES, E. C.; NAVINER, L. A. B.; CAI H. “Single-Event Transient Effects on Dynamic Comparator in 28nm FDSOI CMOS Technology”. In Microeletronics Reliability Journal, vol. 88-90, pp. 965-968, 2018. doi: 10.1016/j.microrel.2018.07.114.

# Appendix B

# Résumé en Français

## B.1 Introduction

Ce chapitre présente le contexte de cette thèse et décrit la motivation ainsi que les contributions du présent travail.

Au cours des dernières décennies, le développement de solutions technologiques dans le domaine des communications a donné naissance à un nouveau concept de radio appelé Radio logicielle (*Software Defined Radio* - SDR) [1]. Il s'agit d'une technologie émergente dans laquelle des fonctions auparavant matérielles sont devenues définies par logiciel, permettant aux utilisateurs de saisir de nouvelles applications avec une SDR. Cette approche présente plusieurs avantages pour le développement de solutions sans fil dans les communications civiles et militaires, telles que l'interopérabilité des troupes de forces et de nations différentes, la portabilité des *waveforms* et la possibilité de mettre à jour les dernières avancées en matière de communications radio sans nécessiter de nouveau matériel.

Un système de communication est essentiellement composé de trois éléments : émetteur, canal et récepteur, comme indiqué sur la Fig. B.1 [2]. Les canaux de communication peuvent être à l'origine de plusieurs distorsions sur le signal, telles que l'atténuation, la perte par trajets multiples et le décalage Doppler, entre autres [2]. Par conséquent, pour mieux récupérer le message envoyé par l'émetteur, une estimation du canal peut être utilisée pour compenser ces effets, ce qui permet une démodulation, une égalisation et un décodage du signal précis. En effet, l'estimation de canal de haute qualité est une caractéristique essentielle des systèmes de communication fiables.

L'un des principaux défis des systèmes de communication consiste à fournir des informations précises sur l'état du canal (CSI) au destinataire [3]. Avec l'estimation du CSI, les symboles trans-

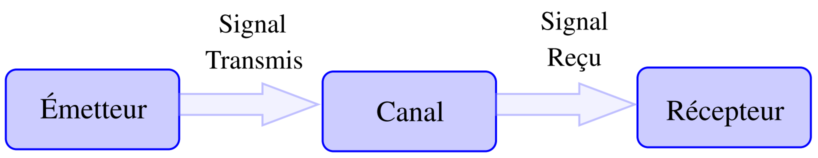


FIGURE B.1 – Eléments d'un système de communication.

mis peuvent être récupérés sur le récepteur. Pour ce faire, plusieurs techniques d'estimation de canal ont été développées afin de fournir le CSI. En général, ces techniques peuvent être classées en trois catégories, comme illustré dans la Fig. B.2 [4].

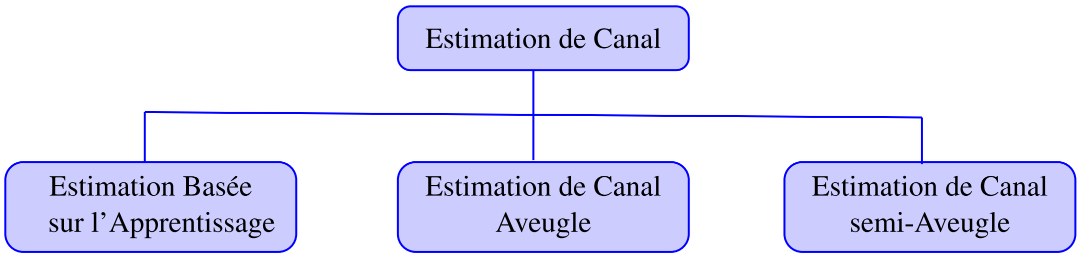

Estimation de Canal
Estimation Basée sur l'Apprentissage
Estimation de Canal Aveugle
Estimation de Canal semi-Aveugle

FIGURE B.2 – Classification des techniques d'estimation de canal.

L'estimation de canal basée sur l'apprentissage (*training based channel estimation*) utilise certains symboles connus, la séquence d'apprentissage, envoyés avant et/ou pendant la communication pour estimer le canal des sorties mesurées à la séquence d'apprentissage. La Fig. B.3 illustre le vecteur de symbole transmis avec $T$ symboles connus (séquence d'apprentissage) et $D$ symboles pour representer l'information (les données). Cependant, ces symboles connus consomment de l'énergie et de la bande passante qui ont une incidence sur l'efficacité spectrale du canal en raison de l'utilisation de ressources de communication pour transmettre ces symboles au lieu de transmettre des données. De plus, dans des canaux variant dans le temps, la séquence d'apprentissage doit être transmise périodiquement, ce qui entraîne une perte supplémentaire du débit du canal. Par conséquent, il est intéressant de réduire la quantité requise de symboles d'apprentissage tout en conservant une précision d'estimation suffisante. Pour ce faire, la caractéristique du canal peut être utilisée pour améliorer les performances de son estimateur.

D'autre part, les techniques aveugles (*Blind Channel Estimation*) se concentrent sur les propriétés statistiques ou déterministes du système. Elles n'exploitent pas la connaissance de la séquence d'apprentissage. Le principal avantage des techniques d'estimation de canaux aveugles est la possibilité d'éliminer les séquences d'apprentissage. Néanmoins, en général, ces techniques nécessitent des enregistrements longs, conduisant à un taux de convergence lent [3].

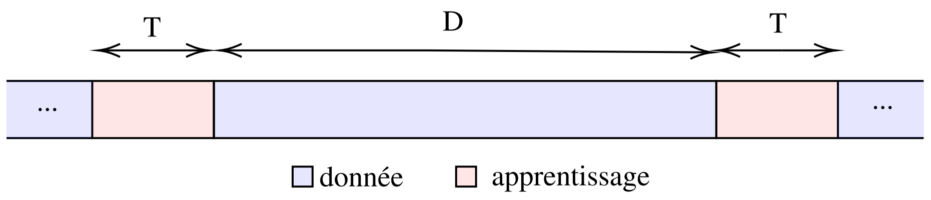


FIGURE B.3 – Vecteur de symbole transmis avec $T$ symboles d'apprentissage et $D$ symboles de données.

Enfin, l'algorithme d'estimation de canal semi-aveugle (*Semiblind Channel Estimation*) est une combinaison d'estimation de canal basée sur la séquence d'apprentissage et l'estimation de canal aveugle. Il utilise les symboles d'apprentissage connus et les symboles de données inconnus pour effectuer une estimation de canal.

Un canal de communication est généralement modélisé par sa réponse impulsionnelle de canal (CIR), qui est un vecteur dont les éléments représentent les gains complexes associés à chaque composant de trajets multiples du canal. Dans des divers systèmes de communication, tels que la télévision haute définition (HDTV) [5, 6], le système mmWave [7, 8], large bande HF [9, 10], bande ultralarge [11–13], massive MIMO [14] et les systèmes de communication sous-marins [15, 16], les canaux peuvent être considérés comme des canaux parcimonieux. En d'autres termes, leurs réponses impulsionnelles sont caractérisées par quelques termes significatifs qui sont largement séparés dans un domaine, c'est-à-dire que de nombreux coefficients sont proches ou égaux à zéro. En effet, dans les systèmes de communication sans fil, par exemple, plusieurs composants du signal arrivent au récepteur avec un retard dû au multitrajet causé par l'environnement, rendant sa réponse parcimonieuse [17].

La longueur d'un canal parcimonieux échantillonné peut atteindre des centaines d'intervalles de symboles, bien que la majorité des taps du canal échantillonné aient une valeur proche de zéro. La parcimonie du canal est définie comme le nombre de éléments non nuls du canal [18, 19].

Les techniques traditionnelles d'estimation de canal telles que les moindres carrés (*least square* - LS) n'exploitent pas la parcimonie des canaux, ce qui entraîne une paramétrisation excessive et des performances médiocres de l'estimateur de canal parcimonieux [2]. En outre, les algorithmes d'estimation classiques deviennent trop complexes pour traiter ces canaux [20].

La caractéristique de parcimonie du signal peut être utilisée pour réduire le coût et la complexité de son estimation. Par exemple, en utilisant l'acquisition comprimée (*compressive sensing* - CS), ces signaux peuvent être reconstruits à partir de moins de mesures que celles requises par le théorème d'échantillonnage de Shannon-Nyquist [18, 21–24]. Le grand avantage de CS est qu'il permet de numériser uniquement les informations du signal pertinentes à un taux d'échantillonnage beaucoup plus bas que le taux de Nyquist, de sorte que le signal numérisé est déjà sous une forme compressée. L'acquisition comprimée a été appliquée à l'estimation des canaux parcimonieux [8, 15, 18, 20, 25–27]. La structure des canaux parcimonieux peut être exploitée à l'aide d'algorithme d'acquisition comprimée tels que Matching Pursuit (MP) [28], Basis Pursuit (BP) [29] et autres [30] conduisant à une meilleure performances d'estimation.

L'estimation de canal peut être formulée comme un problème d'optimisation pouvant être résolu en utilisant des algorithmes numériques qui affinent sa solution de manière itérative. En règle générale, dans l'estimation du canal, une solution précise doit être trouvée avec un petit nombre d'itérations. De plus, ces algorithmes nécessitent des ajustements de paramètres qui, s'ils ne sont pas bien choisis, peuvent réduire les performances de l'algorithme.

Afin de résoudre certains problèmes liés à ces algorithmes, les réseaux de neurones (NNs) sont de plus en plus utilisés dans les systèmes de communication [31–42]. Certaines d'entre eux sont basées sur des algorithmes offrant des garanties de performance et les outils de NN, combinant le meilleur des deux.

À partir de l'étude de l'acquisition comprimée, des algorithmes d'acquisition comprimée et de réseau neuronal, cette thèse vise à contribuer au développement d'algorithmes pour l'estimation de canaux parcimonieux. Dans ce travail, l'estimation de canal est formulée comme un problème de récupération de signal. Les approches développées sont destinées à être utilisées dans des applications radio logicielle (SDR). Cependant, elles peuvent être appliquées dans d'autres domaines où le signal d'intérêt peut être considéré comme parcimonieux ou compressible.

## B.2 Concepts de Base

Cette section introduit quelques concepts importants pour la compréhension de cette thèse. Premièrement, certaines caractéristiques de la radio logicielle sont abordées. Ensuite, le signal parcimonieux et la parcimonie sont expliqués. Enfin, les concepts clés sur l'acquisition comprimée sont présentés et leurs applications illustrées.

**Radio Logicielle (SDR)** En utilisant des périphériques radio traditionnels basés sur du matériel, une intervention physique est nécessaire pour les modifier, limitant ainsi les fonctionnalités croisées et les coûts de production plus élevés. En revanche, la radio logicielle (SDR) apparaît comme une excellente alternative offrant une solution efficace et relativement peu coûteuse à ce problème [43].

Le forum SDR définit un SDR comme “une radio dans laquelle certaines ou toutes les fonctions de la couche physique sont définies par le logiciel” [43]. En d’autres termes, un SDR est une radio implémentée avec un matériel générique qui peut être programmé pour transmettre et recevoir une variété de formes d’onde (*waveform*).

**Signaux Parcimonieux** Des signaux parcimonieux peuvent être trouvés dans plusieurs domaines [46]. Par exemple, la plupart des images et des signaux audio présentent une décomposition parcimonieuse sur une base d’ondelettes et de fréquences-temps, respectivement. Ces signaux sont caractérisés par la concentration d’une grande partie de son énergie dans une petite fraction de sa durée. C’est-à-dire que peu de leurs coefficients ont des valeurs non nulles, ce qui concentre les informations pertinentes.

La Fig. B.4 montre 200 échantillons d’un signal dans le domaine temporel (Fig. B.4a) représentant 8 sinusoides distinctes (Fig. B.4b). Ceci est un exemple d’un signal avec parcimonie égale à 8 representé dans le domaine fréquentiel, c’est-à-dire que, comme indiqué sur la Fig. B.4b, il n’existe que 8 valeurs non nulles parmi les 200 fréquences.

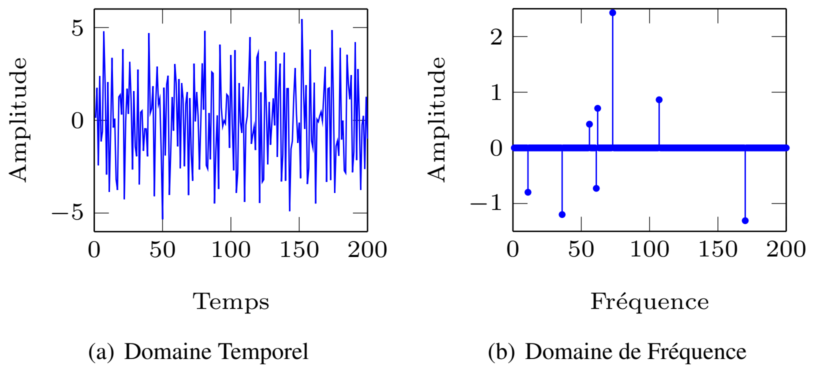


FIGURE B.4 – Échantillons de 8 sinusoides dans le domaine (a) temporel et (b) fréquentiel.

**Acquisition Comprimée** Combinant les processus d’acquisition et de compression et exploitant la parcimonie des signaux, l’acquisition comprimée permet de récupérer le signal parcimonieux à partir d’un nombre de mesures inférieur au minimum requis par le théorème de Nyquist.

Pour ce faire, le signal d’intérêt $\mathbf{z}$ de taille $N$ est d’abord représenté sous la forme d’une combinaison linéaire de seulement $s$ vecteurs de base, avec $s \ll N$, c’est-à-dire $\mathbf{z} = \Psi\mathbf{h}$, où $\mathbf{h}$ est le signal parcimonieux. Ensuite, $\mathbf{z}$ est mesuré en l’échantillonnant par rapport à une matrice $\Phi \in \mathbb{C}^{M\times N}$ :

$$\mathbf{y} = \Phi\mathbf{z} + \mathbf{n} = \Phi\Psi\mathbf{h} + \mathbf{n} = \mathbf{A}\mathbf{h} + \mathbf{n} \tag{B.1}$$

où $\mathbf{y}$ indique le signal reçu de taille $M$ ($M < N$) et $\mathbf{n}$ est le bruit.

Enfin, le signal est reconstruit par un algorithme d'acquisition comprimée [30].

## B.3 Algorithmes d'Acquisition Comprimée

Cette section traite de plusieurs algorithmes d'acquisition comprimée qui ont été proposés au cours des dernières années. Ces algorithmes doivent récupérer un signal parcimonieux à partir d'un ensemble de mesures sous-échantillonné. Ils sont généralement classés en trois catégories principales : relaxation convexe (*convex relaxation*), techniques d'optimisation non convexe (*non-convex optimization techniques*) et algorithmes gloutons (*greedy algorithms*) [24].

La Fig. B.5 montre les algorithmes qui sont abordés dans ce chapitre. Après leur descriptions, certaines comparaisons de performances sont analysées.

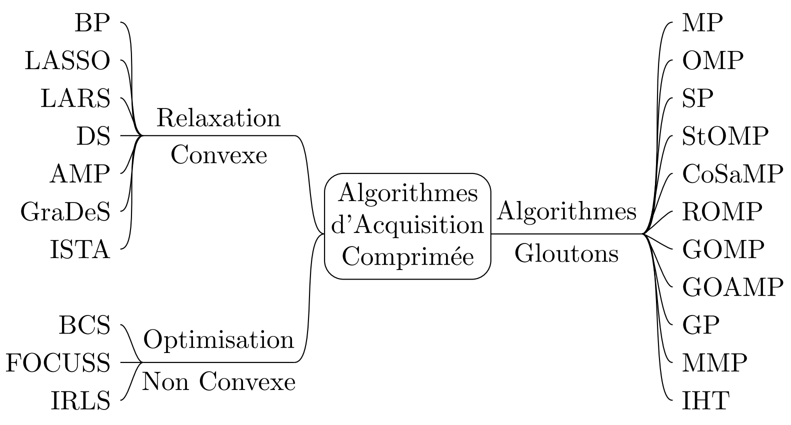


FIGURE B.5 – Classification des algorithmes d'acquisition comprimée.

Les algorithmes de la première catégorie (relaxation convexe) entraînent des problèmes d'optimisation convexe dont les solutions efficaces reposent sur des techniques avancées, telles que les méthodes de gradient projeté, les méthodes de point intérieur ou le seuillage itératif [18].

D'autre part, les approches d'optimisation non convexe peuvent récupérer le signal en tenant compte d'une connaissance préalable de sa distribution [57]. Grâce à une fonction de densité de probabilité postérieure, ces solutions offrent des statistiques complètes de l'estimation. Néanmoins, ils peuvent ne pas convenir aux problèmes de grandes dimensions en raison de leurs exigences informatiques intensives [128].

La troisième catégorie est composée des algorithmes "gloutons". Ils récupèrent le signal de manière itérative, en effectuant une sélection optimale locale à chaque itération dans le but de trouver la solution optimale globale à la fin de l'algorithme.

L'étude de ces algorithmes a abouti à la proposition d'un algorithme glouton appelé *Matching Pursuit based on Least Squares* (MPLS) présenté dans la Section 5.1. De plus, les bons résultats de FISTA et la simplicité de ISTA ont stimulé l'étude du réseau neural LISTA et d'autres réseaux neuronaux utilisés pour récupérer de signaux parcimonieux (voir le Chapitre 4).

## B.4 Apprentissage Profond basé sur l'Acquisition Comprimée

Cette section explore certains réseaux de neurones utilisés pour estimer les signaux parcimonieux. Tout d'abord, les concepts clés liés à l'apprentissage en profondeur sont introduits. Ensuite, plusieurs réseaux de neurones proposés dans la littérature sont abordés. Certaines d'entre eux ont été inspirés par les algorithmes d'acquisition comprimée présentés dans le Chapitre 3.

La Fig. B.6 montre les réseaux de neurones qui sont abordés plus en détail dans ce chapitre. Les algorithmes d'acquisition comprimée qui les ont inspirés sont également illustrés dans cette figure.

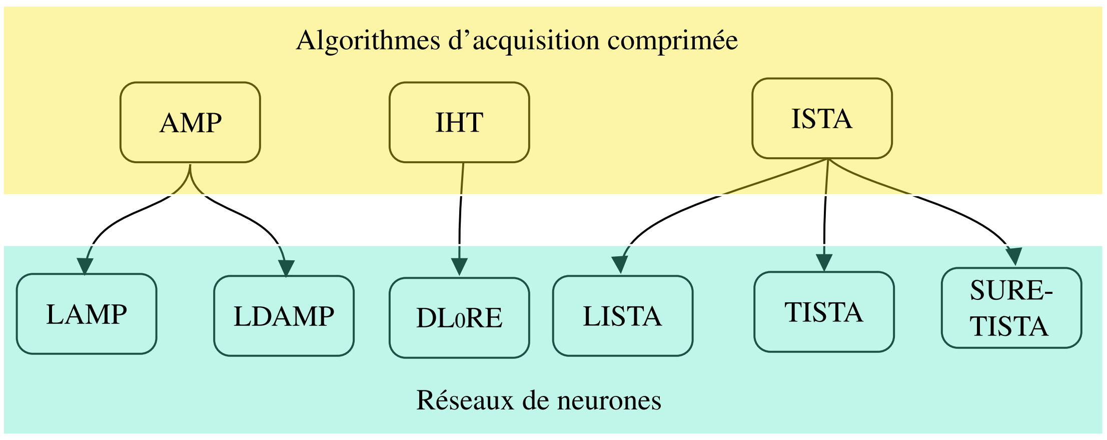


FIGURE B.6 – Les réseaux de neurones inspirés par des algorithmes d'acquisition comprimée présentés dans ce chapitre.

Les réseaux de neurones abordés sont :

- *Learned Iterative Shrinkage-Thresholding Algorithm* (LISTA) ;
- *Learned Approximate Message Passing* (LAMP) ;

- *Deep $l_0$-Regularized Encoder* (D$l_0$RE) ;
- *Learned Denoising-based Approximate Message Passing* (LDAMP) ;
- *Trainable ISTA* (TISTA) ;
- *Stein's Unbiased Risk Estimate based-Trainable Iterative Thresholding Algorithm* (SURE-TISTA).

Dans [200], les auteurs proposent d'utiliser les B-*splines* cubiques comme fonction de retrait dans LISTA. Alternativement, les auteurs de [209] modélisent la fonction d'activation non linéaire en utilisant une expansion linéaire des seuils. Dans cette thèse, d'autres fonctions d'activation à utiliser dans LISTA sont analysées dans la Section 5.5.

L'étude de LISTA a abouti à la proposition de deux alternatives pour améliorer ses performances d'estimation. La première consiste à utiliser l'estimation de LISTA comme premier résultat pour estimer les positions des éléments non nuls d'un signal parcimonieux, puis de calculer leurs valeurs par un processus de pseudo-inversion (voir la Section 5.3). La seconde consiste à remplacer la fonction $\eta_{st}$ utilisée dans LISTA par d'autres fonctions (voir la Section 5.5). En outre, l'étude des réseaux de neurones abordé dans ce chapitre a motivé la proposition d'un nouveau réseau de neurones présentée dans la Section 5.4.

## B.5 Techniques d'Amélioration de l'Estimation de Signaux Parcimonieux

Cette section présente les techniques d'amélioration de l'estimation de signal parcimonieux proposées dans cette thèse. Dans la Section 5.1, un algorithme glouton appelé *Matching Pursuit based on Least Squares* (MPLS) est expliqué. Les algorithmes LS, MP, OMP et MPLS sont utilisés pour estimer deux canaux large bande HF dans la Section 5.2. Ensuite, une alternative pour améliorer les performances d'estimation de LISTA est abordée dans la Section 5.3. Un nouveau réseau de neurones pour l'estimation de signaux parcimonieux est présenté dans la Section 5.4. La Section 5.5 propose des fonctions alternatives pour remplacer la fonction traditionnelle utilisée dans LISTA. Enfin, la Section 5.6 analyse les performances de toutes les techniques proposées dans cette thèse.

**Matching Pursuit based on Least Squares (MPLS)** L'algorithme MPLS s'appuie sur l'algorithme MP pour rechercher les positions des éléments non nuls. Comme pour l'algorithme MP, chaque itération de l'algorithme MPLS consiste à rechercher la colonne $\mathbf{a}_{k_i} \in \mathbf{A}$ qui s'aligne le mieux avec le résidu vector $\mathbf{b}_{i-1}$ ($\mathbf{b}_0 = \mathbf{y}$) et la sélection est effectuée en fonction de :

$$k_i = \arg\max_{l} |\mathbf{a}_l{}^H \mathbf{b_{i-1}}|, \;\; l = 1, 2, ...., N \tag{B.2}$$

La différence entre les algorithmes MP et MPLS réside dans la manière dont les valeurs affectées aux éléments du signal non nuls sont calculées. L'algorithme MP estime le signal sous forme de valeurs de projection, tandis que l'algorithme MPLS effectue cette estimation via l'algorithme LS à la fin de l'algorithme.

Considérons $\Lambda_i$ l'ensemble d'index des colonnes les mieux alignées de $\mathbf{A}$ jusqu'à l'itération $i$, $\Lambda_i$ est actualisé selon les règles suivantes :

- Si $k_i \notin \Lambda_{i-1}$, l'ensemble d'index est mis à jour en tant que $\Lambda_i = \Lambda_{i-1} \cup k_i$.
- Autrement, $\Lambda_i = \Lambda_{i-1}$.

Ensuite, la projection du vecteur résiduel dans cette direction est supprimée et un nouveau vecteur résiduel est calculé comme (B.3).

$$\mathbf{b}_i = \mathbf{b}_{i-1} - P_{\mathbf{a}_{k_i}} \mathbf{b}_{i-1} = \mathbf{b}_{i-1} - \frac{(\mathbf{a}_{k_i}{}^H \mathbf{b}_{i-1}) \mathbf{a}_{k_i}}{||\mathbf{a}_{k_i}||_2^2} \tag{B.3}$$

Après avoir atteint le critère d'arrêt, le signal est estimé par (B.4), où $T$ est le nombre d'itérations et $\mathbf{A}(\Lambda_T)$ est une sous-matrice de $\mathbf{A}$ composé des colonnes $\mathbf{a}_i$ avec $i \in \Lambda_T$.

$$\hat{\mathbf{h}} = \mathbf{A}^{\dagger}(\Lambda_T)\mathbf{y} \tag{B.4}$$

Les performances des algorithmes MP, MPLS et OMP pour l'estimation de canaux parcimonieux ont été évaluées et comparées aux performances théoriques liées à OLS.

L'erreur quadratique moyenne (*Mean-Square Deviation* - MSD) décrite par (B.5) est utilisé pour évaluer les algorithmes en termes de variation de sa parcimonie ($S$), où $N_s$ correspond au nombre de simulations de canaux.

$$MSD = \frac{1}{N_s} \sum ||\mathbf{h} - \hat{\mathbf{h}}||_2^2 \tag{B.5}$$

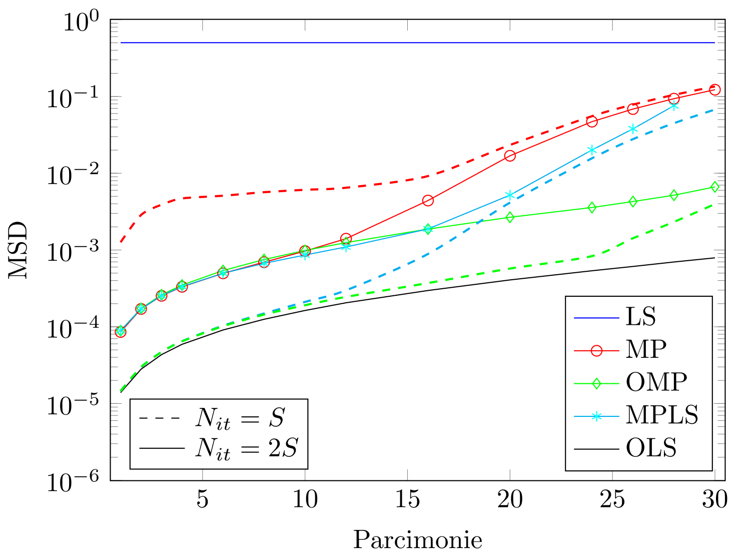


FIGURE B.7 – MSD vs Parcimonie pour $SNR = 10dB$, $M = 60$.

Le modèle du système est défini par (B.6). $N = 120$, $M = 60$, $N_s = 5000$, $SNR = 10\,\text{dB}$, $\mathbf{A}$ représente la matrice de mesure $M \times N$. $\mathbf{A}$ est une matrice de Toeplitz déterminée par la séquence d'apprentissage $\mathbf{c} = [c_1, c_2, ..., c_M]^T$ où $c_i$ est une valeur QPSK. $\mathbf{n}$ est le bruit additif blanc gaussien (*Additive White Gaussian Noise* - AWGN).

$$\mathbf{y} = \mathbf{A}\mathbf{h} + \mathbf{n} \tag{B.6}$$

La Fig. B.7 présente les performances des algorithmes LS, MP, MPLS, OMP et OLS en termes de MSD versus la parcimonie du canal. Soit $N_{it}$ le nombre d'itérations (critère d'arrêt), dans la Fig. 5.1, les lignes continues représentent les simulations avec $N_{it} = 2S$, tandis que les lignes pointillées désignent les simulations avec $N_{it} = S$.

Lorsque $N_{it} = 2S$, les algorithmes MPLS et OMP ont des résultats similaires en termes de MSD jusqu'à 16 éléments non nuls du canal, mais il convient de noter que l'algorithme MPLS obtient ces résultats avec une complexité inférieure à celle de l'algorithme OMP. Pour les algorithmes $S < 10$, les résultats de MP, OMP et MPLS sont similaires. Par conséquent, pour cette range de valeurs $S$, l'algorithme MP est préférable en raison de sa complexité inférieure.

Notez que, pour les petites valeurs de $S$ et $N_{it} = S$, les MSD des algorithmes OMP et MPLS sont proches de ceux obtenus par OLS. Cela signifie que les deux algorithmes trouvent presque toutes les positions correctes des éléments non nuls du canal. De plus, en comparant les résultats pour $N_{it} = 2S$ et $N_{it} = S$, on peut constater que lorsque le critère d'arrêt correspond exactement au nombre de éléments non nuls du canal ($N_{it} = S$), le MSD des deux algorithmes OMP et MPLS décroît contrairement à ce qui se passe avec l'algorithme MP.

Etant donné que les algorithmes MP et MPLS choisissent les positions des éléments non nuls de la même manière, les différentes valeurs de MSD obtenues sont probablement dues à la valeur affectée aux éléments non nuls du canal. D'autre part, l'affectation de la valeur de projection à la valeur d'estimation de canal (effectuée par l'algorithme MP) a un effet très dommageable sur l'estimation de canal. Cela ne se produit pas avec l'algorithme MPLS, car il trouve les positions correctes des éléments non nuls du canal et estime leurs valeurs à l'aide de l'algorithme LS, se rapprochant ainsi des résultats obtenus par OLS.

**Application de l'acquision comprimée pour l'estimation des canaux large bande HF** Cette section analyse l'estimation des canaux large bande HF à l'aide des algorithmes MPLS, MP et OMP. Les résultats suggèrent que les canaux large bande HF pourraient être considérés comme parcimonieux dans le domaine de propagation du retard. Des simulations ont montré que l'utilisation d'un algorithme d'acquisition comprimée prenant en compte la parcimonie du canal permet d'obtenir de meilleures performances d'estimation que d'autres qui ne tiennent pas compte de la parcimonie du canal.

**Proposition visant à améliorer les performances de LISTA** La Fig. B.8 montre le schéma général de l'approche proposée, appelée dans cette thèse comme "Tech. 1" [226].

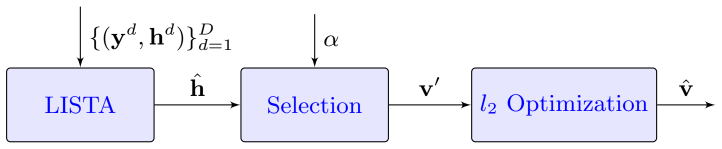


FIGURE B.8 – Schéma général de "Tech. 1".

La première étape consiste en l'implémentation de LISTA, qui produit l'estimation $\hat{\mathbf{h}}$. La deuxième étape ("Selection") prend en compte $\alpha$ et $\hat{\mathbf{h}}$ pour produire le vecteur $\mathbf{v}'$. Cette étape vise à utiliser le seuil $\mu$. Soit $g$ le nombre de taps non nuls de $\hat{\mathbf{h}}$, $E$ la plus grande valeur absolue d'un composant de $\hat{\mathbf{h}}$ et $\alpha$ soit une valeur positive ($\mu = \alpha E$), les éléments du vecteur $\mathbf{v}'$ sont calculés par :

$$\begin{aligned} v'(i) &= 1, \quad \text{si} \quad \hat{h}(i) > \mu \\ v'(i) &= 0, \quad \text{si} \quad \hat{h}(i) \leq \mu \end{aligned} \tag{B.7}$$

C’est-à-dire que cette étape génère un signal $\mathbf{v}'$ avec $j$ éléments 1 où $j \leq g \leq N$. Les positions de ces $j$ éléments non nuls seront les mêmes positions des éléments non nuls de $\hat{\mathbf{v}}$.

Soit $\mathbf{G}(\mathbf{x})$ la matrice $M \times K$ composée uniquement de $K$ colonnes de la matrice $\mathbf{A}$ corrélée aux positions du non-nul éléments de $\mathbf{x}$. La fonction $U(\mathbf{x})$ est définie par :

$$U(\mathbf{x}) = \mathbf{G}^{\dagger}(\mathbf{x})\mathbf{y} = \mathbf{G}^{T}(\mathbf{x})(\mathbf{G}(\mathbf{x})\mathbf{G}^{T}(\mathbf{x}))^{-1}\mathbf{y} \tag{B.8}$$

Enfin, l’étape “$l_2$ Optimization” calcule les éléments non nuls de $\hat{\mathbf{v}}$ par $U(\mathbf{v}')$. Ensuite, l’estimation $\hat{\mathbf{v}}$ du signal fragmenté $\mathbf{h}$ est calculée en attribuant les valeurs du vecteur généré par $U(\mathbf{v}')$ dans le positions $i$ où $v'(i) = 1$.

Le NMSE décrite par (5.2) est utilisé pour évaluer la proposition d’amélioration des performances de LISTA (“Tech. 1”). Le modèle du système est défini par (B.6). $N = 120$, $M = 60$, $N_s = 5000$, $SNR = 30\,\text{dB}$, $\mathbf{A}$ est i.i.d gaussien $\mathcal{N}(0, M^{-1})$. $\mathbf{h}$ est Bernoulli-Gaussian. Ses éléments sont i.i.d $\mathcal{N}(0,1)$ avec probabilité $\gamma = 0,1$ et les autres sont définis comme 0.

Les paramètres de LISTA ont été directement tirés de réalisations indépendantes des données d’apprentissage au cours de la phase d’apprentissage. Les ensembles de données d’apprentissage $\{(\mathbf{y}^d, \mathbf{h}^d)\}_{d=1}^{D}$ avec $D = 10000$ ont été utilisés. Un autre ensemble d’une taille de 10000 a été utilisé pour la phase de test. Il a été généré indépendamment des données d’apprentissage mais à partir de la même distribution. Au cours de la phase d’apprentissage, LISTA adapte les paramètres en minimisant la fonction de perte (B.9) à l’aide de l’optimiseur Adam [197].

$$L_T(\Theta) = \frac{1}{D}\sum_{d=1}^{D} ||\hat{\mathbf{h}}_T(\mathbf{y}^d; \Theta) - \mathbf{h}^d||_2^2 \tag{B.9}$$

La Fig. B.9 présente le NMSE obtenu en fonction du nombre de couches $N_L$ de LISTA. Comme prévu, l’augmentation du nombre de couches entraîne la diminution de la valeur de NMSE. De plus, sur la Fig. B.9, on peut constater qu’en utilisant les étapes proposées à la Fig. B.8, on peut améliorer les performances de l’estimation (diminution de la valeur de NMSE) [226].

Comme le montre la Fig. B.9, avec $\alpha = 0.02$, la valeur de NMSE peut diminuer jusqu’à 5.9 dB par rapport à LISTA$_{\text{std}}$. Cela se produit car, même si LISTA choisit les bons paramètres $\Theta$, l’estimation $\hat{\mathbf{h}}$ donnée par LISTA a plus de elements non nuls que $\mathbf{h}$, conduisant à de plus mauvaises valeurs NMSE. Par conséquent, après l’étape “Sélection”, le nombre de positions des éléments non nuls diminue et l’étape “$l_2$ Optmization” calcule leurs valeurs à l’aide de l’algorithme LS, ce qui entraîne une valeur inférieure de NMSE.

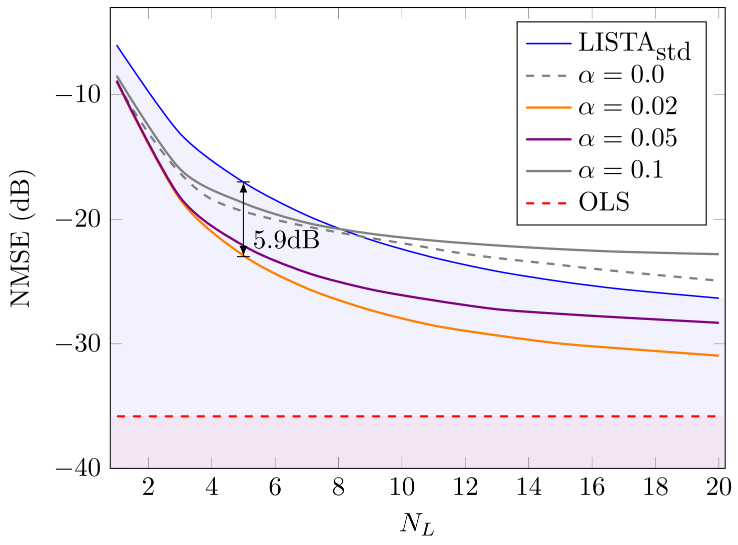


FIGURE B.9 – Estimation de la performance de “Tech. 1”avec différentes valeurs de $\alpha$.

De plus, il ressort de la Fig. B.9 que la valeur $\alpha$ influence beaucoup les performances de l'estimation. Une valeur élevée de $\alpha$ (par exemple, $\alpha = 0.1$) peut entraîner à des nombreuses éléments nuls, ce qui donne une estimation $\hat{\mathbf{v}}$ avec une parcimonie $s$ inférieur à $\mathbf{h}$, générant ainsi de plus grandes valeurs NMSE. D'autre part, une petite valeur $\alpha$ (par exemple $\alpha = 0.0$) peut considérer qu'un grand nombre des éléments ont des valeurs non nulles, ce qui donne une estimation dont la parcimonie $s$ est supérieure à $\mathbf{h}$.

Ces résultats montrent que l'application des deux étapes proposées après la sortie de LISTA peut conduire à de meilleures estimations (valeurs inférieures de NMSE) et moins de couches du réseau de neurones que de considérer directement la sortie donnée par LISTA comme l'estimation du signal d'intérêt. Toutefois, même si ces étapes permettent de mieux estimer le signal parcimonieux, le seuil utilisé peut définir une valeur zéro pour un élément qui doit avoir une valeur non nulle. Dans cette optique, un nouveau réseau de neurones est proposé et décrit dans la Section 5.4 afin d'améliorer l'estimation de signaux parcimonieux axée sur la recherche des positions des éléments non nuls.

Les étapes proposées dans la Fig. B.8 peuvent être appliquées à d'autres réseaux de neurones. Par exemple, il est possible de changer la première étape “LISTA” pour un autre réseau de neurones, par exemple LAMP et d'utiliser l'estimation donnée par LAMP sous la forme $\hat{\mathbf{h}}$ dans la Fig. B.8.

**Réseau de neurones proposé** Cette section décrit le réseau de neurones proposé pour l'estimation de signaux parcimonieux. La Fig. B.10 illustre le réseau de neurones proposé avec $N_L$ couches. Dans cette thèse, ce réseau s'appelle également "Tech. 2" [226]. Le but de ce réseau est d'estimer les positions des éléments non nuls du signal parcimonieux au lieu d'estimer directement les éléments eux-mêmes. Une fois que les positions des éléments non nuls sont estimées par le réseau proposé, leurs valeurs sont calculées en résolvant le problème des moindres carrés. La sortie $\hat{\mathbf{p}}_i$ de la couche $i$ du réseau proposé est donnée par :

$$\hat{\mathbf{p}}_i = \eta_r(\mathbf{S}\hat{\mathbf{p}}_{i-1} + \mathbf{C}\mathbf{y}; \lambda_i) \tag{B.10}$$

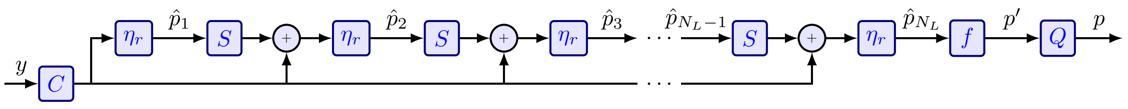


FIGURE B.10 – Réseau de neurones proposé avec $N_L$ couches.

De la même manière que LISTA, les matrices $\mathbf{C}$ et $\mathbf{S}$ et les seuils $\lambda_i$ sont "apprises" pendant la phase d'apprentissage. Cependant, alors que dans LISTA les données d'apprentissage sont composées de signaux parcimonieux $\mathbf{h}^d$, les données d'apprentissage dans le réseau de neurones proposé sont $\{(\mathbf{y}^d, \mathbf{t}^d)\}_{d=1}^{D}$ où $t^d(j) = 1$ si $h^d(j) \neq 0$ et $t^d(j) = 0$ sinon. La fonction de perte utilisée dans le réseau de neurones proposé est donnée par (B.11) où $\Theta = [\mathbf{C}, \mathbf{S}, \lambda]$ et $\lambda = [\lambda_1, \lambda_2, ..., \lambda_{N_L}]$.

$$L(\Theta) = \frac{1}{D}\sum_{d=1}^{D} ||\mathbf{p}'(\mathbf{y}^d; \Theta) - \mathbf{t}^d||_2^2 \tag{B.11}$$

Dans cette thèse, deux fonctions $\eta_r$ différentes sont utilisées : $\eta_{st}$ et $\eta_{pwl}$. La première, $\eta_r = \eta_{st}$, est la même fonction que celle utilisée dans LISTA et définie par (3.16). Par contre, $\eta_r = \eta_{pwl}$ est défini par (B.12), où $\theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$ [227].

$$\eta_{pwl}(x;\theta) = \begin{cases} \theta_3 x, & \text{si} \quad |x| \leq \theta_1 \\ sgn(x)[\theta_4(|x| - \theta_1) + \theta_3\theta_1], & \text{si} \quad \theta_1 < |x| \leq \theta_2 \\ sgn(x)[\theta_5(|x| - \theta_2) + \theta_4(\theta_2 - \theta_1) + \theta_3\theta_1], & \text{si} \quad \theta_2 < |x| \end{cases} \tag{B.12}$$

Après la dernière couche du réseau de neurones proposé, deux autres étapes sont effectuées (voir Fig. B.10) :

- $f$ : c'est la fonction bornée définie par (B.13). Cette fonction a été choisie car elle est une fonction symétrique et ne génère que les valeurs $\in [0,1]$, ce qui entraîne $p'(i) \in [0,1]$.

$$f(x) = |tanh(x)| = |\frac{e^x - e^{-x}}{e^x + e^{-x}}| \tag{B.13}$$

- $Q$ : le vecteur $\mathbf{p}'$ est transformé en vecteur $\mathbf{p}$ composé uniquement d'éléments 0 et 1. Soit $\tau \in [0,1]$, donc :

$$\begin{aligned} p(i) &= 1, \quad \text{si} \quad p'(i) \geq \tau \\ p(i) &= 0, \quad \text{si} \quad p'(i) < \tau \end{aligned} \tag{B.14}$$

Après le réseau de neurones proposé, $\mathbf{p}$ est l'estimation des positions des éléments non nuls de $\hat{\mathbf{h}}$. En d'autres termes, $p(i) = 0$ correspond aux taps de $\hat{\mathbf{h}}$ qui ont une valeur 0 et $p(i) = 1$ représente les taps de $\hat{\mathbf{h}}$ qui ont une valeur non nulle.

Les éléments non nuls de $\hat{\mathbf{h}}$ sont calculés via la solution des moindres carrés. En d'autres termes, les éléments non nuls de l'estimation $\hat{\mathbf{h}}$ du signal parcimonieux $\mathbf{h}$ sont calculés par $U(\mathbf{p})$ (voir (5.23)) et composent le vecteur $\mathbf{u}$. Ensuite, l'estimation $\hat{\mathbf{h}}$ est calculée en attribuant les valeurs du vecteur $\mathbf{u}$ aux positions des éléments non nuls de $\mathbf{p}$.

Les principales différences entre LISTA et le réseau de neurones proposé sont [226] :

- Alors que LISTA estime directement $\hat{\mathbf{h}}$, le réseau de neurones proposé estime les positions des éléments non nuls de $\hat{\mathbf{h}}$.
- Pour estimer les positions des éléments non nuls, deux autres “étapes” sont introduites après la dernière couche du réseau (voir la Fig. 5.13).
- Les valeurs d'éléments de $\hat{\mathbf{h}}$ ne sont pas directement calculées par le réseau de neurones proposé. Ils sont calculés via la solution des moindres carrés basée sur la sortie du réseau de neurones proposé.

Les résultats montrent que le réseau de neurones proposé améliore les valeurs de NMSE par rapport à LISTA. En effet, en utilisant $\eta_r = \eta_{pwl}$ la valeur de NMSE peut diminuer jusqu'à 10.8 dB par rapport à LISTA$_{\text{std}}$ et les résultats obtenus sont proches de OLS. D'autre part, lorsque $\eta_r = \eta_{st}$, Tech. 2 obtient des résultats similaires en termes de NMSE par rapport à celui obtenu avec Tech. 1. De plus, l'impact de la variation statistique du signal parcimonieux utilisé pendant la phase d'apprentissage et la phase d'évaluation est analysé.

**Fonctions alternatives pour LISTA** Cette section explore les fonctions alternatives pour remplacer la fonction traditionnelle $\eta_{st}$ utilisée dans LISTA. L'objectif est d'analyser leur impact en termes de qualité d'estimation. Trois fonctions contrôlées par un petit nombre de paramètres sont prises en compte ici :

***Exponential*** **($\eta_{exp}$) :** Soit $\theta = \{\theta_1, \theta_2, \theta_3\}$, la fonction exponentielle est définie par [227] :

$$\eta_{exp}(x;\theta) = \theta_2 x + \theta_3 x exp\left(-\frac{x^2}{2\theta_1^2}\right) \tag{B.15}$$

***Piecewise Linear*** **($\eta_{pwl}$) :** Soit $\theta = \{\theta_1, \theta_2, \theta_3, \theta_4, \theta_5\}$, cette fonction a cinq segments [227] :

$$\eta_{pwl}(x;\theta) = \begin{cases} \theta_3 x, & \text{si} \quad |x| \leq \theta_1 \\ sgn(x)[\theta_4(|x| - \theta_1) + \theta_3\theta_1], & \text{si} \quad \theta_1 < |x| \leq \theta_2 \\ sgn(x)[\theta_5(|x| - \theta_2) + \theta_4(\theta_2 - \theta_1) + \theta_3\theta_1], & \text{si} \quad \theta_2 < |x| \end{cases} \tag{B.16}$$

***Spline*** **($\eta_{spl}$) :** La fonction *spline* est définie par :

$$\eta_{spl}(x;\theta) = \theta_2 x + \theta_3 x \beta\left(\frac{x}{\theta_1}\right) \tag{B.17}$$

où $\theta = \{\theta_1, \theta_2, \theta_3\}$ et $\beta$ est la fonction cubic B-*spline* [228] :

$$\beta(z) = \begin{cases} \frac{2}{3} - |z|^2 + \frac{|z|^3}{2}, & \text{si} \quad 0 \leq |z| \leq 1 \\ \frac{1}{6}(2 - |z|)^3, & \text{si} \quad 1 \leq |z| \leq 2 \\ 0, & \text{si} \quad 2 \leq |z| \end{cases} \tag{B.18}$$

Les résultats montrent que :

- Les valeurs de NMSE obtenues par LISTA avec les fonctions *piecewise linear*, *exponential*, et *spline* sont meilleures que celles obtenues par LISTA avec la fonction traditionnelle $\eta_{st}$.
- L'utilisation de fonctions *piecewise linear*, *exponential*, et *spline* conduit à une diminution de la valeur de NMSE allant jusqu'à 9 dB par rapport à la fonction de réduction classique.
- L'utilisation d'une des fonctions présentées dans la Section 5.5 permet à LISTA d'obtenir des performances proches de OLS dans un nombre raisonnable de couches.

De plus, la performance d'estimation est analysée lorsque le signal à estimer a des caractéristiques différentes de celles des données d'apprentissage utilisées pour entraîner le réseau. Deux cas différents ont été simulés : le signal à estimer et les signaux utilisés au cours de la phase d'apprentissage ont des valeurs différentes de SNR ; et le signal à estimer et les signaux utilisés pendant la phase d'apprentissage ont différentes valeurs $\gamma$.

Les résultats indiquent que le choix de la fonction de retrait est un facteur important pour la performance d'estimation de LISTA. En effet, des fonctions de retrait plus flexibles, telles qu'*exponential*, *spline*, ou *piecewise linear* ont conduit à de meilleures performances d'estimation que la fonction de retrait traditionnelle $\eta_{st}$. La fonction *piecewise linear* présente un léger avantage par rapport aux autres fonctions de retrait lorsque les caractéristiques de l'ensemble d'apprentissage et le signal à estimer sont identiques (Fig. 5.20). D'autre part, en termes de robustesse à la non-concordance de SNR, les fonctions *spline* et *exponential* présentent un léger avantage (Fig. 5.21).

**Comparaison de performance** Cette section présente une comparaison de performance entre certains algorithmes d'acquisition comprimée et les propositions de cette thèse. Tout d'abord, l'implémentation arithmétique en virgule flottante est considérée (voir la Section 5.6.1). Les performances d'estimation sont ensuite analysées à l'aide de l'arithmétique en virgule fixe (voir la Section 5.6.2).

Pour l'implémentation arithmétique en virgule flottante, les résultats indiquent que PNN$_{pwl}$ est la meilleure technique pour estimer un signal parcimonieux. Elle a le plus petit pourcentage de taps ayant une attribution incorrecte et atteint une valeur de NMSE proche de celles de OLS. Même si LISTA$_{pwl}$ conduit à des valeurs NMSE légèrement inférieures à celles de PNN$_{pwl}$, LISTA$_{pwl}$ a l'inconvénient d'attribuer des valeurs non nulles à un grand nombre de taps qui devraient avoir une valeur nulle, ce qui donne un gros $P_e$.

Cependant, les performances des techniques varient lorsque le nombre de bits utilisés dans la partie fractionnaire change. De cette simulation, LISTA$_s$ est la technique qui permet d'obtenir de meilleurs valeurs de NMSE pour $Q8.8$ tandis que pour $Q8.16$, le PNN$_{pwl}$ conduit au plus bas pourcentage de taps avec une attribution incorrecte et de bonnes valeurs NMSE.

## B.6 Conclusion et Perspectives

Cette section comprend quelques remarques finales sur le travail actuel et les perspectives.

Cette thèse portait sur l'estimation des canaux parcimonieux et l'exploration de nouvelles approches pour l'estimation des signaux parcimonieux. Une estimation de canal de haute qualité est très importante pour un système de communication fiable. Plusieurs canaux peuvent être considérés comme parcimonieux. Cette caractéristique peut être utilisée pour améliorer leur estimation. L'acquisition comprimée et ses algorithmes sont utilisés dans plusieurs domaines et peuvent être appliqués à une estimation de canal parcimonieux réduisant le coût et la complexité de l'estimation ainsi que le nombre de mesures nécessaires. Le travail effectué au cours de cette thèse s'est focalisé principalement sur deux aspects : les algorithmes d'acquisition comprimée et les réseaux de neurones basés sur l'acquisition comprimée.

Partant de la revue de littérature, plusieurs algorithmes d'acquisition comprimée ont été étudiés et analysés dans le Chapitre 3. Une analyse de performance suggère que ces algorithmes permettent de mieux estimer les signaux parcimonieux. En d'autres termes, plus le signal est parcimonieux, plus l'erreur d'estimation sera petit. De plus, les algorithmes BCS et OMP présentaient les valeurs les plus basses de NMSE (voir la Section 3.4). Cette étude a conduit à la première contribution de cette thèse : la proposition d'un algorithme appelé *Matching Pursuit based on Least Squares* (MPLS) qui combine les algorithmes *Matching Pursuit* et *Least Square* (voir la Section 5.1). Les résultats ont montré que l'algorithme MPLS offre un meilleur compromis entre la précision de l'estimation et le coût de calcul que l'algorithme OMP, tout en produisant des résultats similaires.

L'étude de ces algorithmes a également suscité l'intérêt et l'étude des réseaux de neurones inspirés par certains de ces algorithmes. Ils sont discutés dans le Chapitre 4. Trois autres contributions de cette thèse ont été présentées dans les Sections 5.3, 5.4 et 5.5 afin d'améliorer la récupération de signaux parcimonieux.

LISTA peut être considéré comme l'un des premiers réseaux de neurones inspiré d'un algorithme d'acquisition comprimée. Dans cette thèse, il est proposé d'utiliser la sortie LISTA pour donner une direction à la sélection des positions des éléments non nuls du signal. Ces positions sont choisies en fonction d'un seuil lié au plus grand élément donné par la sortie de LISTA. Ensuite, l'estimation du signal sera calculée par un processus de pseudo-inversion (voir la Section 5.3). Les résultats obtenus montrent qu'en utilisant cette approche, la valeur de NMSE peut être réduite jusqu'à 5.9 dB par rapport aux performances d'estimation de LISTA.

De plus, un nouveau réseau de neurones axé sur la recherche des positions des éléments non nuls a été proposé (voir la Section 5.4). Les résultats obtenus montrent que le réseau proposé est une bonne approche pour la récupération de signaux parcimonieux en réduisant jusqu'à 10.8 dB la valeur NMSE par rapport à LISTA et en obtenant des performances proches de OLS. En outre, il faut moins de couches pour obtenir de bonnes estimations, ce qui réduit le temps nécessaire à la phase de formation et réduit le nombre de multiplications matrice-vecteur nécessaires pendant la phase d'évaluation.

Par ailleurs, d'autres fonctions de retrait sont proposées pour remplacer la fonction de retrait traditionnelle de LISTA. Leurs performances d'estimation sont analysées dans la Section 5.5. Les résultats obtenus indiquent que les estimations générées par le réseau LISTA nécessitent moins de multiplications matrice-vecteur que les algorithmes existants dotés de paramètres de régularisation optimisés. De plus, en choisissant une fonction de retrait adéquate, LISTA peut atteindre des valeurs de NMSE proches des performances théoriques liées à quelques couches.

Les résultats de cette thèse renforcent l'adéquation du réseau de neurones à l'estimation de signaux parcimonieux comme alternative aux algorithmes d'acquisition comprimée. Les approches développées dans ce travail peuvent être étendues et utilisées pour l'estimation de signaux parcimonieux dans plusieurs domaines tels que ceux présentés dans la Section 2.3.2.

La plupart des travaux couverts par cette thèse ont été publiés dans des conférences et un journal du domaine. Ces publications sont répertoriées dans l'annexe A.

# Bibliography

[1] J. Mitola, “SDR architecture refinement for JTRS,” in *MILCOM 2000 Proceedings. 21st Century Military Communications. Architectures and Technologies for Information Superiority (Cat. No.00CH37155)*, vol. 1, Oct 2000, pp. 214–218 vol.1, doi: 10.1109/MILCOM.2000.904942.

[2] S. Haykin, *Adaptive filter theory*, 4th ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[3] O. O. Oyerinde and S. H. Mneney, “Review of channel estimation for wireless communication systems,” *IETE Technical Review*, vol. 29, no. 4, pp. 282–298, 2012, doi: 10.4103/0256-4602.101308.

[4] K.-L. Du and M. Swamy, *Wireless communication systems: From RF subsystems to 4G enabling technologies*, 01 2010, doi: 10.1017/CBO9780511841453.

[5] W. F. Schreiber, “Advanced television systems for terrestrial broadcasting: Some problems and some proposed solutions,” *Proceedings of the IEEE*, vol. 83, no. 6, pp. 958–981, Jun 1995, doi: 10.1109/5.387095.

[6] Z. Fan, Z. Lu, and Y. Han, “Accurate channel estimation based on bayesian compressive sensing for next-generation wireless broadcasting systems,” in *2014 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, June 2014, pp. 1–5, doi: 10.1109/BMSB.2014.6873548.

[7] Z. Marzi, D. Ramasamy, and U. Madhow, “Compressive channel estimation and tracking for large arrays in mm-Wave picocells,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 3, pp. 514–527, April 2016, doi: 10.1109/JSTSP.2016.2520899.

[8] X. Ma, F. Yang, S. Liu, J. Song, and Z. Han, “Design and optimization on training sequence for mmWave communications: A new approach for sparse channel estimation in massive

MIMO," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 7, pp. 1486–1497, July 2017, doi: 10.1109/JSAC.2017.2698978.

[9] J. Ying, J. Zhong, M. Zhao, and Y. Cai, "Turbo equalization based on compressive sensing channel estimation in wideband HF systems," in *2013 International Conference on Wireless Communications and Signal Processing*, Oct 2013, pp. 1–5, doi: 10.1109/WCSP.2013.6677272.

[10] E. C. Marques, N. Maciel, L. A. B. Naviner, H. Cai, and J. Yang, "Compressed sensing for wideband HF channel estimation," *International Conference on Frontiers of Signal Processing*, pp. 1–5, 2018, doi: 10.1109/ICFSP.2018.8552050.

[11] P. Zhang, Z. Hu, R. C. Qiu, and B. M. Sadler, "A compressed sensing based ultra-wideband communication system," in *2009 IEEE International Conference on Communications*, June 2009, pp. 1–5, doi: 10.1109/ICC.2009.5198584.

[12] K. M. Cohen, C. Attias, B. Farbman, I. Tselniker, and Y. C. Eldar, "Channel estimation in UWB channels using compressed sensing," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1966–1970, doi: 10.1109/ICASSP.2014.6853942.

[13] S. Sharma, A. Gupta, and V. Bhatia, "A new sparse signal-matched measurement matrix for compressive sensing in UWB communication," *IEEE Access*, vol. 4, pp. 5327–5342, 2016, doi: 10.1109/ACCESS.2016.2601779.

[14] X. Rao and V. K. N. Lau, "Compressive sensing with prior support quality information and application to massive MIMO channel estimation with temporal correlation," *IEEE Transactions on Signal Processing*, vol. 63, no. 18, pp. 4914–4924, Sep. 2015, doi: 10.1109/TSP.2015.2446444.

[15] E. Panayirci, H. Senol, M. Uysal, and H. V. Poor, "Sparse channel estimation and equalization for OFDM-based underwater cooperative systems with amplify-and-forward relaying," *IEEE Transactions on Signal Processing*, vol. 64, no. 1, pp. 214–228, Jan 2016, doi: 10.1109/TSP.2015.2477807.

[16] C. Li, K. Song, and L. Yang, "Low computational complexity design over sparse channel

estimator in underwater acoustic OFDM communication system," *IET Communications*, vol. 11, no. 7, pp. 1143–1151, 2017, doi: 10.1049/iet-com.2016.1215.

[17] Y. C. Eldar and G. Kutyniok, *Compressed Sensing: Theory and Applications*, 1st ed. Cambridge University Press, 2012.

[18] C. R. Berger, Z. Wang, J. Huang, and S. Zhou, "Application of compressive sensing to sparse channel estimation," *IEEE Communications Magazine*, vol. 48, no. 11, pp. 164–174, November 2010, doi: 10.1109/MCOM.2010.5621984.

[19] J. W. Choi, B. Shim, Y. Ding, B. Rao, and D. I. Kim, "Compressed sensing for wireless communications: Useful tips and tricks," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1527–1550, 2017, doi: 10.1109/COMST.2017.2664421.

[20] C. Carbonelli, S. Vedantam, and U. Mitra, "Sparse channel estimation with zero tap detection," *IEEE Transactions on Wireless Communications*, vol. 6, no. 5, pp. 1743–1763, May 2007, doi: 10.1109/TWC.2007.360376.

[21] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, Feb 2006, doi: 10.1109/TIT.2005.862083.

[22] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006, doi: 10.1109/TIT.2006.871582.

[23] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 21–30, March 2008, doi: 10.1109/MSP.2007.914731.

[24] A. Y. Carmi, L. S. Mihaylova, and S. J. Godsill, *Compressed Sensing & Sparse Filtering*. Springer, 2014.

[25] S. F. Cotter and B. D. Rao, "Sparse channel estimation via matching pursuit with application to equalization," *IEEE Transactions on Communications*, vol. 50, no. 3, pp. 374–377, Mar 2002, doi: 10.1109/26.990897.

[26] H. qian Huang, W. Su, and X. lin Jiang, "An improved compressed sensing reconstruction algorithm used in sparse channel estimation," in *2016 IEEE International Conference on*

*Signal Processing, Communications and Computing (ICSPCC)*, Aug 2016, pp. 1–4, doi: 10.1109/ICSPCC.2016.7753737.

[27] G. Z. Karabulut and A. Yongacoglu, "Sparse channel estimation using orthogonal matching pursuit algorithm," in *IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. 2004*, vol. 6, Sept 2004, pp. 3880–3884 Vol. 6, doi: 10.1109/VETECF.2004.1404804.

[28] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, Dec 1993, doi: 10.1109/78.258082.

[29] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, pp. 33–61, 1998.

[30] E. C. Marques, N. Maciel, L. Naviner, H. Cai, and J. Yang, "A review of sparse recovery algorithms," *IEEE Access*, vol. 7, pp. 1300–1322, 2019, doi: 10.1109/ACCESS.2018.2886471.

[31] M. Borgerding, P. Schniter, and S. Rangan, "AMP-inspired deep networks for sparse linear inverse problems," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4293–4308, Aug 2017, doi: 10.1109/TSP.2017.2708040.

[32] H. He, C. Wen, S. Jin, and G. Y. Li, "Deep learning-based channel estimation for beamspace mmWave massive MIMO systems," *IEEE Wireless Communications Letters*, vol. 7, no. 5, pp. 852–855, Oct 2018, doi: 10.1109/LWC.2018.2832128.

[33] ——, "A model-driven deep learning network for MIMO detection," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Nov 2018, pp. 584–588, doi: 10.1109/GlobalSIP.2018.8646357.

[34] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," *Proceedings of the 27th International Conference on International Conference on Machine Learning*, p. 399–406, 2010.

[35] M. Borgerding and P. Schniter, "Onsager-corrected deep learning for sparse linear inverse problems," in *2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec 2016, pp. 227–231, doi: 10.1109/GlobalSIP.2016.7905837.

[36] Z. Wang, Q. Ling, and T. S. Huang, "Learning deep $l_0$ encoders," *Proceedings of the thirtieh AAAI Conference on Artificial Intelligence*, p. 2194–2200, 2016.

[37] C. Metzler, A. Mousavi, and R. Baraniuk, "Learned D-AMP: Principled neural network based compressive image recovery," in *Advances in Neural Information Processing Systems 30*, 2017, pp. 1772–1783.

[38] P. Sprechmann, A. M. Bronstein, and G. Sapiro, "Learning efficient sparse and low rank models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1821–1833, Sep. 2015, doi: 10.1109/TPAMI.2015.2392779.

[39] M. Yao, J. Dang, Z. Zhang, and L. Wu, "SURE-TISTA: A signal recovery network for compressed sensing," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2019, pp. 3832–3836, doi: 10.1109/ICASSP.2019.8683182.

[40] D. Ito, S. Takabe, and T. Wadayama, "Trainable ISTA for sparse signal recovery," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2018, pp. 1–6, doi: 10.1109/ICCW.2018.8403660.

[41] ——, "Trainable ISTA for sparse signal recovery," *IEEE Transactions on Signal Processing*, vol. 67, no. 12, pp. 3113–3125, June 2019, doi: 10.1109/TSP.2019.2912879.

[42] S. Takabe and T. Wadayama, "Complex field-trainable ISTA for linear and nonlinear inverse problems," *CoRR*, vol. abs/1904.07409, 2019. [Online]. Available: http://arxiv.org/abs/1904.07409

[43] W. I. Forum, "What is software defined radio?" https://www.wirelessinnovation.org/what_is_sdr, [Online; accessed 19-July-2017].

[44] T. F. Collins, R. Getz, D. Pu, and A. M. Wyglinski, *Software-Defined Radio for Engineers*. Artech House Publishers, 2018.

[45] C. R. Aguayo Gonzalez, C. B. Dietrich, and J. H. Reed, "Understanding the software communications architecture," *IEEE Communications Magazine*, vol. 47, no. 9, pp. 50–57, Sep. 2009, doi: 10.1109/MCOM.2009.5277455.

[46] A. M. Bruckstein, D. L. Donoho, and M. Elad, “From sparse solutions of systems of equations to sparse modeling of signals and images,” *SIAM Rev.*, vol. 51, no. 1, pp. 34–81, Feb. 2009.

[47] E. Candes, J. Romberg, and T. Tao, “Stable signal recovery from incomplete and inaccurate measurements,” *Communications on Pure and Applied Mathematics*, vol. 59, no. 8, pp. 1207–1223, 2006, doi: 10.1002/cpa.20124.

[48] E. J. Candes and T. Tao, “Near-optimal signal recovery from random projections: Universal encoding strategies?” *IEEE Transactions on Information Theory*, vol. 52, no. 12, pp. 5406–5425, Dec 2006, doi: 10.1109/TIT.2006.885507.

[49] ——, “Decoding by linear programming,” *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, Dec 2005, doi: 10.1109/TIT.2005.858979.

[50] C. M. Verdun, “Compressive sensing,” Master’s thesis, Universidade Federal do Rio de Janeiro, 2016.

[51] G. R. de Baron de Prony, “Essai expérimental et analytique: Sur les lois de la dilatabilité de fluides élastique et sur celles de la force expansive de la vapeur de l’alcool, à différentes températures,” *Journal de l’École Polytechnique Floréal et Plairial*, vol. 1, no. 22, pp. 24–76, 1795.

[52] S. Levy and P. Fullagar, “Reconstruction of a sparse spike train from a portion of its spectrum and application to high-resolution deconvolution,” *Geophysics*, vol. 46, p. 1235, Sep. 1981, doi: 10.1190/1.1441261.

[53] C. Walker and T. J. Ulrych, “Autoregressive recovery of the acoustic impedance,” *Geophysics*, vol. 48, no. 10, pp. 1338–1350, 10 1983, doi: 10.1190/1.1441414.

[54] F. Santosa and W. Symes, “Linear inversion of band-limited reflection seismograms,” *SIAM Journal on Scientific and Statistical Computing*, vol. 7, no. 4, pp. 1307–1330, 1986, doi: 10.1137/0907087.

[55] S. S. Chen and D. L. Donoho, “Basis pursuit,” *Technical Report, Depart- ment of Statistics, Stanford University*, 1994.

[56] D. L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2845–2862, Nov 2001, doi: 10.1109/18.959265.

[57] Y. Arjoune, N. Kaabouch, H. E. Ghazi, and A. Tamtaoui, "Compressive sensing: Performance comparison of sparse recovery algorithms," in *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, Jan 2017, pp. 1–7, doi: 10.1109/CCWC.2017.7868430.

[58] E. Candes and J. Romberg, "Sparsity and incoherence in compressive sampling," *Inverse Problems*, vol. 23, no. 3, pp. 969–985, 2007, doi: 10.1088/0266-5611/23/3/008.

[59] M. Abo-Zahhad, A. Hussein, and A. Mohamed, "Compressive sensing algorithms for signal processing applications: A survey," *International Journal of Communications, Network and System Sciences*, vol. 8, pp. 197–216, 2015, doi: 10.4236/ijcns.2015.86021.

[60] J. Wen, D. Li, and F. Zhu, "Stable recovery of sparse signals via $l_p$-minimization," *Applied and Computational Harmonic Analysis*, vol. 38, no. 1, pp. 161 – 176, 2015, doi: 10.1016/j.acha.2014.06.003.

[61] R. G. Baraniuk, "Compressive sensing [lecture notes]," *IEEE Signal Processing Magazine*, vol. 24, no. 4, pp. 118–121, July 2007, doi: 10.1109/MSP.2007.4286571.

[62] Z. Chen and J. J. Dongarra, "Condition numbers of gaussian random matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 27, no. 3, pp. 603–620, Jul. 2005, doi: 10.1137/040616413.

[63] W. U. Bajwa, J. D. Haupt, G. M. Raz, S. J. Wright, and R. D. Nowak, "Toeplitz-structured compressed sensing matrices," in *2007 IEEE/SP 14th Workshop on Statistical Signal Processing*, Aug 2007, pp. 294–298, doi: 10.1109/SSP.2007.4301266.

[64] A. Amini, V. Montazerhodjat, and F. Marvasti, "Matrices with small coherence using $p$-ary block codes," *IEEE Transactions on Signal Processing*, vol. 60, no. 1, pp. 172–181, Jan 2012, doi: 10.1109/TSP.2011.2169249.

[65] R. Calderbank, S. Howard, and S. Jafarpour, "Construction of a large class of deterministic sensing matrices that satisfy a statistical isometry property," *IEEE Journal of Selected Top-*

*ics in Signal Processing*, vol. 4, no. 2, pp. 358–374, April 2010, doi: 10.1109/JSTSP.2010.2043161.

[66] S. Li, F. Gao, G. Ge, and S. Zhang, "Deterministic construction of compressed sensing matrices via algebraic curves," *IEEE Transactions on Information Theory*, vol. 58, no. 8, pp. 5035–5041, Aug 2012, doi: 10.1109/TIT.2012.2196256.

[67] T. Nguyen and Y. Shin, "Deterministic sensing matrices in compressive sensing: A survey," *The Scientific World Journal*, 2013, doi: 10.1155/2013/192795.

[68] M. Elad and A. M. Bruckstein, "A generalized uncertainty principle and sparse representation in pairs of bases," *IEEE Transactions on Information Theory*, vol. 48, no. 9, pp. 2558–2567, Sep 2002, doi: 10.1109/TIT.2002.801410.

[69] D. L. Donoho and Y. Tsaig, "Fast solution of $l_1$-norm minimization problems when the solution may be sparse," *IEEE Transactions on Information Theory*, vol. 54, no. 11, pp. 4789–4812, Nov 2008, doi: 10.1109/TIT.2008.929958.

[70] D. L. Donoho and J. Tanner, "Counting faces of randomly projected polytopes when the projection radically lowers dimension," *Journal of the American Mathematical Society*, vol. 22, no. 1, pp. 1–53, 2009.

[71] G. Satat, M. Tancik, and R. Raskar, "Lensless imaging with compressive ultrafast sensing," *IEEE Transactions on Computational Imaging*, vol. 3, no. 3, pp. 398–407, Sept 2017, doi: 10.1109/TCI.2017.2684624.

[72] U. V. Dias and M. E. Rane, "Block based compressive sensed thermal image reconstruction using greedy algorithms," *International Journal of Image, Graphics and Signal Processing*, vol. 6, no. 10, pp. 36–42, 2014, doi: 10.5815/ijigsp.2014.10.05.

[73] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baraniuk, "Single-pixel imaging via compressive sampling," *IEEE Signal Processing Magazine*, vol. 25, no. 2, pp. 83–91, March 2008, doi: 10.1109/MSP.2007.914730.

[74] A. Saucedo, S. Lefkimmiatis, N. Rangwala, and K. Sung, "Improved computational efficiency of locally low rank MRI reconstruction using iterative random patch adjustments," *IEEE Transactions on Medical Imaging*, vol. 36, no. 6, pp. 1209–1220, June 2017, doi: 10.1109/TMI.2017.2659742.

[75] F. Pareschi, P. Albertini, G. Frattini, M. Mangia, R. Rovatti, and G. Setti, "Hardware-algorithms co-design and implementation of an analog-to-information converter for biosignals based on compressed sensing," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 1, pp. 149–162, Feb 2016, doi: 10.1109/TBCAS.2015.2444276.

[76] S. Vasanawala, M. Murphy, M. Alley, P. Lai, K. Keutzer, J. Pauly, and M. Lustig, "Practical parallel imaging compressed sensing MRI: Summary of two years of experience in accelerating body MRI of pediatric patients," in *2011 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, March 2011, pp. 1039–1043, doi: 10.1109/ISBI.2011.5872579.

[77] D. Craven, B. McGinley, L. Kilmartin, M. Glavin, and E. Jones, "Adaptive dictionary reconstruction for compressed sensing of ECG signals," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 3, pp. 645–654, May 2017, doi: 10.1109/JBHI.2016.2531182.

[78] H. Djelouat, X. Zhai, M. A. Disi, A. Amira, and F. Bensaali, "System-on-chip solution for patients biometric: A compressive sensing-based approach," *IEEE Sensors Journal*, pp. 1–1, 2018, doi: 10.1109/JSEN.2018.2871411.

[79] S. Pudlewski, A. Prasanna, and T. Melodia, "Compressed-sensing-enabled video streaming for wireless multimedia sensor networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 6, pp. 1060–1072, Jun. 2012, doi: 10.1109/TMC.2011.175.

[80] B. Srinivasarao, V. C. Gogineni, S. Mula, and I. Chakrabarti, "A novel framework for compressed sensing based scalable video coding," *Signal Processing: Image Communication*, vol. 57, pp. 183–196, 2017, doi: 10.1016/j.image.2017.05.002.

[81] C. Li, H. Jiang, P. Wilford, Y. Zhang, and M. Scheutzow, "A new compressive video sensing framework for mobile broadcast," *IEEE Transactions on Broadcasting*, vol. 59, no. 1, pp. 197–205, March 2013, doi: 10.1109/TBC.2012.2226509.

[82] T. Goldstein, L. Xu, K. F. Kelly, and R. Baraniuk, "The STOne transform: Multi-resolution image enhancement and compressive video," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 5581–5593, Dec 2015, doi: 10.1109/TIP.2015.2474697.

[83] R. G. Baraniuk, T. Goldstein, A. C. Sankaranarayanan, C. Studer, A. Veeraraghavan, and M. B. Wakin, "Compressive video sensing: Algorithms, architectures, and applications,"

*IEEE Signal Processing Magazine*, vol. 34, no. 1, pp. 52–66, Jan 2017, doi: 10.1109/MSP.2016.2602099.

[84] K. V. Siddamal, S. P. Bhat, and V. S. Saroja, "A survey on compressive sensing," in *2015 2nd International Conference on Electronics and Communication Systems (ICECS)*, Feb 2015, pp. 639–643, doi: 10.1109/ECS.2015.7124986.

[85] M. Herman and T. Strohmer, "Compressed sensing radar," in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2008, pp. 1509–1512, doi: 10.1109/ICASSP.2008.4517908.

[86] M. S. Kang, S. J. Lee, S. H. Lee, and K. T. Kim, "ISAR imaging of high-speed maneuvering target using gapped stepped-frequency waveform and compressive sensing," *IEEE Transactions on Image Processing*, vol. 26, no. 10, pp. 5043–5056, Oct 2017, doi: 10.1109/TIP.2017.2728182.

[87] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002, doi: 10.1016/S1389-1286(01)00302-4.

[88] M. A. Razzaque, C. Bleakley, and S. Dobson, "Compression in wireless sensor networks: A survey and comparative evaluation," *ACM Trans. Sen. Netw.*, vol. 10, no. 1, pp. 5:1–5:44, Dec. 2013.

[89] C. Karakus, A. C. Gurbuz, and B. Tavli, "Analysis of energy efficiency of compressive sensing in wireless sensor networks," *IEEE Sensors Journal*, vol. 13, no. 5, pp. 1999–2008, May 2013, doi: 10.1109/JSEN.2013.2244036.

[90] M. Hooshmand, M. Rossi, D. Zordan, and M. Zorzi, "Covariogram-based compressive sensing for environmental wireless sensor networks," *IEEE Sensors Journal*, vol. 16, no. 6, pp. 1716–1729, March 2016, doi: 10.1109/JSEN.2015.2503437.

[91] Z. Li, H. Huang, and S. Misra, "Compressed sensing via dictionary learning and approximate message passing for multimedia internet of things," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 505–512, April 2017, doi: 10.1109/JIOT.2016.2583465.

[92] M. Mangia, F. Pareschi, R. Rovatti, and G. Setti, "Low-cost security of IoT sensor nodes with rakeness-based compressed sensing: Statistical and known-plaintext attacks," *IEEE*

*Transactions on Information Forensics and Security*, vol. PP, no. 99, pp. 1–1, 2017, doi: 10.1109/TIFS.2017.2749982.

[93] Y. Gargouri, H. Petit, P. Loumeau, B. Cecconi, and P. Desgreys, “Compressed sensing for astrophysical signals,” in *2016 IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, Dec 2016, pp. 313–316, doi: 10.1109/ICECS.2016.7841195.

[94] J. Bobin, J. L. Starck, and R. Ottensamer, “Compressed sensing in astronomy,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 5, pp. 718–726, Oct 2008, doi: 10.1109/JSTSP.2008.2005337.

[95] J. Yang, “A machine learning paradigm based on sparse signal representation,” Ph.D. dissertation, University of Wollongong, 2013.

[96] J. Lu, N. Verma, and N. K. Jha, “Compressed signal processing on nyquist-sampled signals,” *IEEE Transactions on Computers*, vol. 65, no. 11, pp. 3293–3303, Nov 2016, doi: 10.1109/TC.2016.2532861.

[97] B. Sun, H. Feng, K. Chen, and X. Zhu, “A deep learning framework of quantized compressed sensing for wireless neural recording,” *IEEE Access*, vol. 4, pp. 5169–5178, 2016, doi: 10.1109/ACCESS.2016.2604397.

[98] S. K. Sharma, E. Lagunas, S. Chatzinotas, and B. Ottersten, “Application of compressive sensing in cognitive radio communications: A survey,” *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 1838–1860, 2016, doi: 10.1109/COMST.2016.2524443.

[99] H. Sun, A. Nallanathan, C. X. Wang, and Y. Chen, “Wideband spectrum sensing for cognitive radio networks: a survey,” *IEEE Wireless Communications*, vol. 20, no. 2, pp. 74–81, April 2013, doi: 10.1109/MWC.2013.6507397.

[100] A. Ali and W. Hamouda, “Advances on spectrum sensing for cognitive radio networks: Theory and applications,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 2, pp. 1277–1304, 2017, doi: 10.1109/COMST.2016.2631080.

[101] F. Salahdine, N. Kaabouch, and H. E. Ghazi, “A survey on compressive sensing techniques for cognitive radio network,” *Physical Commun. J., Elsevier*, 2016, doi: 10.1016/j.phycom.2016.05.002.

[102] W. U. Bajwa, J. Haupt, A. M. Sayeed, and R. Nowak, "Compressed channel sensing: A new approach to estimating sparse multipath channels," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1058–1076, June 2010, doi: 10.1109/JPROC.2010.2042415.

[103] B. Mansoor, S. J. Nawaz, and S. M. Gulfam, "Massive-MIMO sparse uplink channel estimation using implicit training and compressed sensing," *Applied Sciences*, vol. 7, 2017, doi: 10.3390/app7010063.

[104] R. Ferdian, Y. Hou, and M. Okada, "A low-complexity hardware implementation of compressed sensing-based channel estimation for ISDB-T system," *IEEE Transactions on Broadcasting*, vol. 63, no. 1, pp. 92–102, March 2017, doi: 10.1109/TBC.2016.2617286.

[105] J. A. Tropp, J. N. Laska, M. F. Duarte, J. K. Romberg, and R. G. Baraniuk, "Beyond nyquist: Efficient sampling of sparse bandlimited signals," *IEEE Transactions on Information Theory*, vol. 56, no. 1, pp. 520–544, Jan 2010, doi: 10.1109/TIT.2009.2034811.

[106] X. Chen, Z. Yu, S. Hoyos, B. M. Sadler, and J. Silva-Martinez, "A sub-nyquist rate sampling receiver exploiting compressive sensing," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 3, pp. 507–520, March 2011, doi: 10.1109/TCSI.2010.2072430.

[107] M. Pelissier and C. Studer, "Non-uniform wavelet sampling for rf analog-to-information conversion," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 2, pp. 471–484, Feb 2018, doi: 10.1109/TCSI.2017.2729779.

[108] W. Guo, Y. Kim, A. H. Tewfik, and N. Sun, "A fully passive compressive sensing SAR ADC for low-power wireless sensors," *IEEE Journal of Solid-State Circuits*, vol. 52, no. 8, pp. 2154–2167, Aug 2017, doi: 10.1109/JSSC.2017.2695573.

[109] Y. Zhang, X. Fu, Q. Zhang, Z. Feng, and X. Liu, "Novel schemes to optimize sampling rate for compressed sensing," *Journal of Communications and Networks*, vol. 17, no. 5, pp. 517–524, Oct 2015, doi: 10.1109/JCN.2015.000090.

[110] J. N. Laska, S. Kirolos, M. F. Duarte, T. S. Ragheb, R. G. Baraniuk, and Y. Massoud, "Theory and implementation of an analog-to-information converter using random demodulation," in *2007 IEEE International Symposium on Circuits and Systems*, May 2007, pp. 1959–1962, doi: 10.1109/ISCAS.2007.378360.

[111] T. Ragheb, J. N. Laska, H. Nejati, S. Kirolos, R. G. Baraniuk, and Y. Massoud, “A prototype hardware for random demodulation based compressive analog-to-digital conversion,” in *2008 51st Midwest Symposium on Circuits and Systems*, Aug 2008, pp. 37–40, doi: 10.1109/MWSCAS.2008.4616730.

[112] Y. Chen, M. Mishali, Y. C. Eldar, and A. O. Hero, “Modulated wideband converter with non-ideal lowpass filters,” in *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2010, pp. 3630–3633, doi: 10.1109/ICASSP.2010.5495912.

[113] D. E. Bellasi, L. Bettini, C. Benkeser, T. Burger, Q. Huang, and C. Studer, “VLSI design of a monolithic compressive-sensing wideband analog-to-information converter,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 3, no. 4, pp. 552–565, Dec 2013, doi: 10.1109/JETCAS.2013.2284618.

[114] Y. Gargouri, H. Petit, P. Loumeau, B. Cecconi, and P. Desgreys, “Analog-to-information converter design for low-power acquisition of astrophysical signals,” in *2017 15th IEEE International New Circuits and Systems Conference (NEWCAS)*, June 2017, pp. 113–116, doi: 10.1109/NEWCAS.2017.8010118.

[115] M. Trakimas, R. D’Angelo, S. Aeron, T. Hancock, and S. Sonkusale, “A compressed sensing analog-to-information converter with edge-triggered sar adc core,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 5, pp. 1135–1148, May 2013, doi: 10.1109/TCSI.2013.2244435.

[116] J. F. Gemmeke, H. V. Hamme, B. Cranen, and L. Boves, “Compressive sensing for missing data imputation in noise robust speech recognition,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 4, no. 2, pp. 272–287, April 2010, doi: 10.1109/JSTSP.2009.2039171.

[117] M. Gavrilescu, “Improved automatic speech recognition system by using compressed sensing signal reconstruction based on $l_0$ and $l_1$ estimation algorithms,” in *2015 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, June 2015, doi: 10.1109/ECAI.2015.7301156.

[118] U. P. Shukla, N. B. Patel, and A. M. Joshi, “A survey on recent advances in speech compressive sensing,” in *2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*, March 2013, pp. 276–280, doi: 10.1109/iMac4s.2013.6526422.

[119] A. Latif and W. A. Mousa, "An efficient undersampled high-resolution radon transform for exploration seismic data processing," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 2, pp. 1010–1024, Feb 2017, doi: 10.1109/TGRS.2016.2618848.

[120] J. Cao, Y. Wang, J. Zhao, and C. Yang, "A review on restoration of seismic wavefields based on regularization and compressive sensing," *Inverse Problems in Science and Engineering*, vol. 19, no. 5, p. 679–704, 2011, doi: 10.1080/17415977.2011.576342.

[121] C. Stork and D. Brookes, "The decline of conventional seismic acquisition and the rise of specialized acquisition: This is compressive sensing," *Society of Exploration Geophysicists Annual Meeting*, p. 4386–4392, 2014, doi: 10.1190/segam2014-0870.1.

[122] M. Hawes, L. Mihaylova, F. Septier, and S. Godsill, "Bayesian compressive sensing approaches for direction of arrival estimation with mutual coupling effects," *IEEE Transactions on Antennas and Propagation*, vol. 65, no. 3, pp. 1357–1368, March 2017, doi: 10.1109/TAP.2017.2655013.

[123] Q. Wang, T. Dou, H. Chen, W. Yan, and W. Liu, "Effective block sparse representation algorithm for DOA estimation with unknown mutual coupling," *IEEE Communications Letters*, vol. 21, no. 12, pp. 2622–2625, Dec 2017, doi: 10.1109/LCOMM.2017.2747547.

[124] L. Zhao, J. Xu, J. Ding, A. Liu, and L. Li, "Direction-of-arrival estimation of multipath signals using independent component analysis and compressive sensing," *PLOS ONE*, vol. 12, no. 7, pp. 1–17, 07 2017, doi: 10.1371/journal.pone.0181838.

[125] D. Malioutov, M. Cetin, and A. S. Willsky, "A sparse signal reconstruction perspective for source localization with sensor arrays," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 3010–3022, Aug 2005, doi: 10.1109/TSP.2005.850882.

[126] I. Bilik, T. Northardt, and Y. Abramovich, "Expected likelihood for compressive sensing-based DOA estimation," in *IET International Conference on Radar Systems (Radar 2012)*, Oct 2012, pp. 1–4, doi: 10.1049/cp.2012.1710.

[127] Q. Shen, W. Liu, W. Cui, and S. Wu, "Underdetermined DOA estimation under the compressive sensing framework: A review," *IEEE Access*, vol. 4, pp. 8865–8878, 2016, doi: 10.1109/ACCESS.2016.2628869.

[128] D. Kanevsky, A. Carmi, L. Horesh, P. Gurfil, B. Ramabhadran, and T. N. Sainath, “Kalman filtering for compressed sensing,” in *2010 13th International Conference on Information Fusion*, July 2010, pp. 1–8, doi: 10.1109/ICIF.2010.5711877.

[129] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*, 2nd ed. Norwell, MA: Kluwer, 2001.

[130] P. S. Huggins and S. W. Zucker, “Greedy basis pursuit,” *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3760–3772, July 2007, doi: 10.1109/TSP.2007.894287.

[131] R. Tibshirani, “Regression shrinkage and selection via the LASSO,” *Journal of the Royal Statistical Society, Series B: Methodological*, vol. 58, pp. 267–288, 1996.

[132] W. Li, “Estimation and tracking of rapidly time-varying broadband acoustic communication channels,” Ph.D. dissertation, Massachusetts Institute of Technology, 2006.

[133] W. J. Fu, “Penalized regressions: The bridge versus the LASSO,” *Journal of Computational and Graphical Statistics*, vol. 7, no. 3, pp. 397–416, 1998, doi: 10.1080/10618600.1998.10474784.

[134] A. Maleki, L. Anitori, Z. Yang, and R. G. Baraniuk, “Asymptotic analysis of complex LASSO via complex approximate message passing (CAMP),” *IEEE Transactions on Information Theory*, vol. 59, no. 7, pp. 4290–4308, July 2013, doi: 10.1109/TIT.2013.2252232.

[135] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, “Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, Dec 2007, doi: 10.1109/JSTSP.2007.910281.

[136] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, “Least angle regression,” *Annals of statistics*, vol. 32, no. 2, pp. 407–451, 2004, doi: 10.1214/009053604000000067.

[137] M. A. Hameed, “Comparative analysis of orthogonal matching pursuit and least angle regression,” Master’s thesis, Michigan State University, United State of America, 2012.

[138] E. Candes and T. Tao, “The dantzig selector: statistical estimation when $p$ is much larger than $n$,” *The Annals of Statistics*, vol. 35, no. 6, p. 2313–2351, 2007, doi: 10.1214/009053606000001523.

[139] I. Daubechies, M. Defrise, and C. D. Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” *Communications on Pure and Applied Mathematics*, vol. 57, no. 11, p. 1413–1457, 2004, doi: 10.1002/cpa.20042.

[140] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm with application to wavelet-based image deblurring,” in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2009, pp. 693–696, doi: 10.1109/ICASSP.2009.4959678.

[141] A. Maleki, “Approximate message passing algorithms for compressed sensing,” Ph.D. dissertation, Stanford University, 2011.

[142] D. L. Donoho, A. Maleki, and A. Montanari, “Message passing algorithms for compressed sensing,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, no. 45, pp. 18 914–18 919, 2009, doi: 10.1073/pnas.0909892106.

[143] J. Ma and L. Ping, “Orthogonal AMP,” *IEEE Access*, vol. 5, pp. 2020–2033, 2017, doi: 10.1109/ACCESS.2017.2653119.

[144] R. Garg and R. Khandekar, “Gradient descent with sparsification: An iterative algorithm for sparse recovery with restricted isometry property,” 01 2009, p. 43, doi: 10.1145/1553374.1553417.

[145] D. P. Wipf and B. D. Rao, “Sparse bayesian learning for basis selection,” *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2153–2164, Aug 2004, doi: 10.1109/TSP.2004.831016.

[146] S. Ji, Y. Xue, and L. Carin, “Bayesian compressive sensing,” *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2346–2356, June 2008, doi: 10.1109/TSP.2007.914345.

[147] M. A. T. Figueiredo, “Adaptive sparseness using jeffreys prior,” in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, ser. NIPS’01. Cambridge, MA, USA: MIT Press, 2001, pp. 697–704.

[148] J. M. Bernardo and A. F. M. Smith, “Bayesian theory,” *New York: Wiley*, 1994.

[149] M. E. Tipping, “Sparse bayesian learning and the relevance vector machine,” *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001, doi: 10.1162/15324430152748236.

[150] I. F. Gorodnitsky and B. D. Rao, “Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm,” *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 600–616, Mar 1997, doi: 10.1109/78.558475.

[151] R. Chartrand and W. Yin, “Iteratively reweighted algorithms for compressive sensing,” in *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, March 2008, pp. 3869–3872, doi: 10.1109/ICASSP.2008.4518498.

[152] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, “Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition,” in *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, Nov 1993, pp. 40–44 vol.1, doi: 10.1109/ACSSC.1993.342465.

[153] S. Kwon, J. Wang, and B. Shim, “Multipath matching pursuit,” *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2986–3001, May 2014, doi: 10.1109/TIT.2014.2310482.

[154] W. Dai and O. Milenkovic, “Subspace pursuit for compressive sensing signal reconstruction,” *IEEE Transactions on Information Theory*, vol. 55, no. 5, pp. 2230–2249, May 2009, doi: 10.1109/TIT.2009.2016006.

[155] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, “Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit,” *IEEE Transactions on Information Theory*, vol. 58, no. 2, pp. 1094–1121, Feb 2012, doi: 10.1109/TIT.2011.2173241.

[156] D. Needell and J. A. Tropp, “CoSaMP: Iterative signal recovery from incomplete and inaccurate samples,” California Institute of Technology, Pasadena, Tech. Rep., 2008.

[157] D. Needell and R. Vershynin, “Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit,” *Found. Comput. Math.*, vol. 9, no. 3, pp. 317–334, Apr. 2009, doi: 10.1007/s10208-008-9031-3.

[158] J. Wang, S. Kwon, and B. Shim, “Generalized orthogonal matching pursuit,” *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6202–6216, Dec 2012, doi: 10.1109/TSP.2012.2218810.

[159] H. Sun and L. Ni, “Compressed sensing data reconstruction using adaptive generalized orthogonal matching pursuit algorithm,” in *Proceedings of 2013 3rd International Con-*

*ference on Computer Science and Network Technology*, Oct 2013, pp. 1102–1106, doi: 10.1109/ICCSNT.2013.6967295.

[160] T. Blumensath and M. E. Davies, “Gradient pursuits,” *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2370–2382, June 2008, doi: 10.1109/TSP.2007.916124.

[161] G. Pope, “Compressive sensing: a summary of reconstruction algorithms,” Master’s thesis, Eidgenossische Technische Hochschule, Swiss, 2009.

[162] G. H. Golub and C. F. V. Loan, “Matrix computations,” *MD: Johns Hopkins Univ. Press*, 1996.

[163] C. La and M. Do, “Signal reconstruction using sparse tree representations,” *Proc. Wavelets XI SPIE Opt. Photon.*, pp. 273–283, August 2005, doi: 10.1117/12.621064.

[164] T. Blumensath and M. E. Davies, “Iterative Thresholding for Sparse Approximations,” *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 629–654, 2008, doi: 10.1007/s00041-008-9035-z.

[165] “Iterative hard thresholding for compressed sensing,” *Applied and Computational Harmonic Analysis*, vol. 27, no. 3, pp. 265 – 274, 2009, doi: 10.1016/j.acha.2009.04.002.

[166] H. Huang and A. Makur, “Backtracking-based matching pursuit method for sparse signal reconstruction,” *IEEE Signal Processing Letters*, vol. 18, no. 7, pp. 391–394, July 2011, doi: 10.1109/LSP.2011.2147313.

[167] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, “Algorithmic linear dimension reduction in the $\ell_1$ norm for sparse vectors,” in *Allerton 2006 - 44th Annual Allerton Conference on Communication, Control, and Computing*, 2006.

[168] J. D. Blanchard, J. Tanner, and K. Wei, “CGIHT: conjugate gradient iterative hard thresholding for compressed sensing and matrix completion,” *Information and Inference: A Journal of the IMA*, vol. 4, no. 4, pp. 289–327, 2015, doi: 10.1093/imaiai/iav011.

[169] X. Zhu, L. Dai, W. Dai, Z. Wang, and M. Moonen, “Tracking a dynamic sparse channel via differential orthogonal matching pursuit,” in *MILCOM 2015 - 2015 IEEE Military Communications Conference*, Oct 2015, pp. 792–797, doi: 10.1109/MILCOM.2015.7357541.

[170] C. A. Metzler, A. Maleki, and R. G. Baraniuk, "From denoising to compressed sensing," *IEEE Transactions on Information Theory*, vol. 62, no. 9, pp. 5117–5144, Sep. 2016, doi: 10.1109/TIT.2016.2556683.

[171] N. B. Karahanoglu and H. Erdogan, "Compressed sensing signal recovery via forward–backward pursuit," *Digital Signal Processing*, vol. 23, no. 5, pp. 1539 – 1548, 2013, doi: 10.1016/j.dsp.2013.05.007.

[172] A. C. Gilbert, S. Muthukrishnan, and M. Strauss, "Improved time bounds for near-optimal sparse fourier representations," *Proceedings of the SPIE 5914, Wavelets XI*, vol. 5914, 2005, doi: 10.1117/12.615931.

[173] S. Foucart, "Hard thresholding pursuit: An algorithm for compressive sensing," *SIAM Journal on Numerical Analysis*, vol. 49, no. 6, pp. 2543–2563, 2011, doi: 10.1137/100806278.

[174] A. C. Gilbert, M. J. Strauss, J. A. Tropp, and R. Vershynin, "One sketch for all: Fast algorithms for compressed sensing," in *Proceedings of the Thirty-ninth Annual ACM Symposium on Theory of Computing*, 2007, pp. 237–246, doi: 10.1145/1250790.1250824.

[175] Z. Xu, X. Chang, F. Xu, and H. Zhang, "$l_{1/2}$ regularization: A thresholding representation theory and a fast solver," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1013–1027, July 2012, doi: 10.1109/TNNLS.2012.2197412.

[176] J. Tanner and K. Wei, "Normalized iterative hard thresholding for matrix completion," *SIAM Journal on Scientific Computing*, vol. 35, no. 5, pp. S104–S125, 2013, doi: 10.1137/120876459.

[177] J. K. Pant and S. Krishnan, "Two-pass $l_p$-regularized least-squares algorithm for compressive sensing," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2017, pp. 1–4, doi: 10.1109/ISCAS.2017.8050535.

[178] G. Mileounis, B. Babadi, N. Kalouptsidis, and V. Tarokh, "An adaptive greedy algorithm with application to nonlinear communications," *IEEE Transactions on Signal Processing*, vol. 58, no. 6, pp. 2998–3007, June 2010, doi: 10.1109/TSP.2010.2044841.

[179] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, p. 2479–2493, July 2009, doi: 10.1109/TSP.2009.2016892.

[180] E. Vlachos, A. S. Lalos, and K. Berberidis, “Stochastic gradient pursuit for adaptive equalization of sparse multipath channels,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 3, pp. 413–423, Sept 2012, doi: 10.1109/JETCAS.2012.2214631.

[181] B. A. Olshausen and K. J. Millman, “Learning sparse codes with a mixture-of-gaussians prior,” in *Advances in Neural Information Processing Systems 12*. MIT Press, 2000.

[182] C. Guo and M. E. Davies, “Near optimal compressed sensing without priors: Parametric SURE approximate message passing,” *IEEE Transactions on Signal Processing*, vol. 63, no. 8, pp. 2130–2141, April 2015, doi: 10.1109/TSP.2015.2408569.

[183] J. Lee, J. W. Choi, and B. Shim, “Sparse signal recovery via tree search matching pursuit,” *Journal of Communications and Networks*, vol. 18, no. 5, pp. 699–712, October 2016, doi: 10.1109/JCN.2016.000100.

[184] S. Rangan, P. Schniter, and A. K. Fletcher, “Vector approximate message passing,” *2017 IEEE International Symposium on Information Theory (ISIT)*, vol. abs/1610.03082, pp. 1588–1592, June 2017, doi: 10.1109/ISIT.2017.8006797.

[185] H. Zhao, S. Ding, X. Li, and H. Huang, “Deep neural network structured sparse coding for online processing,” *IEEE Access*, vol. 6, pp. 74 778–74 791, 2018, doi: 10.1109/ACCESS.2018.2882531.

[186] M. Dhasmana and S. Budhiraja, “A survey of compressive sensing based greedy pursuit reconstruction algorithms,” *International Journal of Image, Graphics and Signal Processing (IJIGSP)*, vol. 7, no. 10, 2015, doi: 10.5815/ijigsp.2015.10.01.

[187] L. Bai, P. Maechler, M. Muehlberghuber, and H. Kaeslin, “High-speed compressed sensing reconstruction on FPGA using OMP and AMP,” in *2012 19th IEEE International Conference on Electronics, Circuits, and Systems (ICECS 2012)*, Dec 2012, pp. 53–56, doi: 10.1109/ICECS.2012.6463559.

[188] T. Wang, C. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, “Deep learning for wireless physical layer: Opportunities and challenges,” *China Communications*, vol. 14, pp. 92–111, 2017, doi: 10.1109/CC.2017.8233654.

[189] T. J. O'Shea and J. Hoydis, "An introduction to machine learning communications systems," *CoRR*, vol. abs/1702.00832, 2017.

[190] "DR2-Net: Deep residual reconstruction network for image compressive sensing," *Neurocomputing*, 2019, doi: 10.1016/j.neucom.2019.05.006.

[191] H. Lu and L. Bo, "WDLReconNet: Compressive sensing reconstruction with deep learning over wireless fading channels," *IEEE Access*, vol. 7, pp. 24 440–24 451, 2019, doi: 10.1109/ACCESS.2019.2900715.

[192] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3142–3155, July 2017, doi: 10.1109/TIP.2017.2662206.

[193] S. Ji, W. Xu, M. Yang, and K. Yu, "3D convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, Jan 2013, doi: 10.1109/TPAMI.2012.59.

[194] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, Feb 2016, doi: 10.1109/TPAMI.2015.2439281.

[195] H. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, May 2016, doi: 10.1109/TMI.2016.2528162.

[196] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, vol. 22, no. 3, p. 400–407, 1951, doi: 10.1214/aoms/1177729586.

[197] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of International Conference on Learning Representations*, 2015.

[198] T. Kohonen, "Self-organizing maps," in *Springer Series in Information Sciences*, 1995, doi: 10.1007/978-3-642-97610-0.

[199] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990, doi: 10.1109/5.58325.

[200] U. S. Kamilov and H. Mansour, “Learning optimal nonlinearities for iterative thresholding algorithms,” *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 747–751, May 2016, doi: 10.1109/LSP.2016.2548245.

[201] A. Mousavi and R. G. Baraniuk, “Learning to invert: Signal recovery via deep convolutional networks,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 2272–2276, doi: 10.1109/ICASSP.2017.7952561.

[202] A. Mousavi, A. B. Patel, and R. G. Baraniuk, “A deep learning approach to structured signal recovery,” in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2015, pp. 1336–1343, doi: 10.1109/ALLERTON.2015.7447163.

[203] B. Xin, Y. Wang, W. Gao, D. P. Wipf, and B. Wang, “Maximal sparsity with deep networks?” in *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2016, pp. 4340–4348.

[204] H. Ye, G. Y. Li, and B. Juang, “Power of deep learning for channel estimation and signal detection in OFDM systems,” *IEEE Wireless Communications Letters*, vol. 7, no. 1, pp. 114–117, Feb 2018, doi: 10.1109/LWC.2017.2757490.

[205] L. Li, L. G. Wang, F. L. Teixeira, C. Liu, A. Nehorai, and T. J. Cui, “DeepNIS: Deep neural network for nonlinear electromagnetic inverse scattering,” *IEEE Transactions on Antennas and Propagation*, vol. 67, no. 3, pp. 1819–1825, March 2019, doi: 10.1109/TAP.2018.2885437.

[206] L. Bo, H. Lu, Y. Lu, J. Meng, and W. Wang, “FompNet: Compressive sensing reconstruction with deep learning over wireless fading channels,” in *2017 9th International Conference on Wireless Communications and Signal Processing (WCSP)*, Oct 2017, pp. 1–6, doi: 10.1109/WCSP.2017.8171076.

[207] J. R. Hershey, J. L. Roux, and F. Weninger, “Deep unfolding: Model-based inspiration of novel deep architectures,” *CoRR*, vol. abs/1409.2574, 2014. [Online]. Available: http://arxiv.org/abs/1409.2574

[208] R. F. Molanes, K. Amarasinghe, J. Rodriguez-Andina, and M. Manic, “Deep learning and reconfigurable platforms in the internet of things: Challenges and opportunities in algorithms and hardware,” *IEEE Industrial Electronics Magazine*, vol. 12, no. 2, pp. 36–49, June 2018, doi: 10.1109/MIE.2018.2824843.

[209] D. Mahapatra, S. Mukherjee, and C. S. Seelamantula, “Deep sparse coding using optimized linear expansion of thresholds,” *CoRR*, vol. abs/1705.07290, 2017. [Online]. Available: http://arxiv.org/abs/1705.07290

[210] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[211] C. Trabelsi, O. Bilaniuk, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, “Deep complex networks,” *CoRR*, vol. abs/1705.09792, 2017.

[212] N. M. de Paiva Jr., E. C. Marques, and L. A. de Barros Naviner, “Sparsity analysis using a mixed approach with greedy and LS algorithms on channel estimation,” in *2017 3rd International Conference on Frontiers of Signal Processing (ICFSP)*, 2017, pp. 91–95, doi: 10.1109/ICFSP.2017.8097148.

[213] M. R. Heidarpour and M. Uysal, “Multicarrier HF communications with amplify-and-forward relaying,” in *2011 IEEE 22nd International Symposium on Personal, Indoor and Mobile Radio Communications*, Sept 2011, pp. 1396–1400, doi: 10.1109/PIMRC.2011.6139731.

[214] M. Hervás, J. L. Pijoan, R. Alsina-Pagès, M. Salvador, and D. Altadill, “Channel sounding and polarization diversity for the NVIS channel,” *Proceedings of Nordic HF Conference*, August 2013.

[215] J. Mitola, *Cognitive Radio Architecture: The Engineering Foundations of Radio XML*. Wiley, 2006.

[216] “Stanag 5066: Profile for high frequency (HF) radio data communications,” *North Atlantic Treaty Organization*, 2010.

[217] "Interoperability and performance standards for data modems: MIL-STD-188–110C," *D. O. D Interface Standard*, 2011.

[218] S. A. Laraway, H. Moradi, and B. Farhang-Boroujeny, "HF band filter bank multi-carrier spread spectrum," in *MILCOM 2015 - 2015 IEEE Military Communications Conference*, Oct 2015, pp. 1445–1453, doi: 10.1109/MILCOM.2015.7357648.

[219] J. F. Mastrangelo, J. J. Lemmon, L. E. Vogler, J. A. Hoffmeyer, L. E. Pratt, and C. J. Behm, "A new wideband high frequency channel simulation system," *IEEE Transactions on Communications*, vol. 45, no. 1, pp. 26–34, Jan 1997, doi: 10.1109/26.554283.

[220] D. J. Belknap, R. D. Haggarty, and B. D. Perry, "Adaptive signal processing for ionospheric distortion correction," *Mitre Tech. Rep*, 1968.

[221] R. P. Basler, G. H. Price, R. T. Tsunoda, and T. L. Wong, "HF channel probe," *Proceedings of Ionospheric Effects Symposium*, 1987.

[222] W. N. Furman, J. W. Nieto, and W. M. Batts, "Wideband HF channel availability – measurement techniques and results," *14th International Ionospheric Effects Symposium*, 2015.

[223] C. Watterson, J. Juroshek, and W. Bensema, "Experimental confirmation of an HF channel model," *IEEE Transactions on Communication Technology*, vol. 18, no. 6, pp. 792–803, December 1970, doi: 10.1109/TCOM.1970.1090438.

[224] Z. Yan, L. Zhang, T. Rahman, and D. Su, "Prediction of the HF ionospheric channel stability based on the modified ITS model," *IEEE Transactions on Antennas and Propagation*, vol. 61, no. 6, pp. 3321–3333, June 2013, doi: 10.1109/TAP.2013.2249571.

[225] L. Wagner and J. Goldstein, "Channel spread parameters for the highlatitude, near-vertical-incidence-skywave HF channel: Correlation with geomagnetic activity," *DTiC Document, Tech. Rep.*, 1995.

[226] E. C. Marques, N. Maciel, L. Naviner, H. Cai, and J. Yang, "Deep learning approaches for sparse recovery in compressive sensing," *Proceedings of the 11th International Symposium on Image and Signal Processing and Analysis*, 2019, doi: 10.1109/ISPA.2019.8868841.

[227] C. Guo and M. E. Davies, "Near optimal compressed sensing without priors: Parametric sure approximate message passing," *IEEE Transactions on Signal Processing*, vol. 63, no. 8, pp. 2130–2141, April 2015, doi: 10.1109/TSP.2015.2408569.

[228] M. Unser, “Splines: a perfect fit for signal and image processing,” *IEEE Signal Processing Magazine*, vol. 16, no. 6, pp. 22–38, Nov 1999, doi: 10.1109/79.799930.

[229] E. C. Marques, N. Maciel, L. Naviner, H. Cai, and J. Yang, “Nonlinear functions in learned iterative shrinkage-thresholding algorithm for sparse signal recovery,” *2019 IEEE International Workshop on Signal Processing Systems (SiPS)*, 2019.

[230] L. D. Pyeatt, *Modern Assembly Language Programming with the ARM Processor*. Elsevier Science, 2016, doi: 10.1016/C2015-0-00180-0.

[231] A. M. Kulkarni, H. Homayoun, and T. Mohsenin, “A parallel and reconfigurable architecture for efficient OMP compressive sensing reconstruction,” in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, ser. GLSVLSI ’14. New York, NY, USA: ACM, 2014, pp. 299–304, doi: 10.1145/2591513.2591598. [Online]. Available: http://doi.acm.org/10.1145/2591513.2591598

[232] A. Kulkarni and T. Mohsenin, “Low overhead architectures for OMP compressive sensing reconstruction algorithm,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 6, pp. 1468–1480, June 2017, doi: 10.1109/TCSI.2017.2648854.

[233] S. Salehi, M. B. Mashhadi, A. Zaeemzadeh, N. Rahnavard, and R. F. DeMara, “Energy-aware adaptive rate and resolution sampling of spectrally sparse signals leveraging VCMA-MTJ devices,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 8, no. 4, pp. 679–692, Dec 2018, doi: 10.1109/JETCAS.2018.2857998.

INSTITUT
POLYTECHNIQUE
DE PARIS

**Titre :** Estimation de Canaux Parcimonieux pour la Radio Logicielle

**Mots clés :** Acquisition Comprimée ; Réseau neuronal ; Estimation des Canaux Parcimonieux

**Résumé :** Les canaux de communication sont utilisés pour transmettre des signaux d'information. Cependant, ces canaux peuvent causer plusieurs distorsions sur le signal à transmettre, telles que l'atténuation, la perte par trajets multiples et le décalage Doppler, entre autres. Pour une meilleure récupération des messages, le récepteur peut estimer le canal et améliorer la fiabilité des systèmes de communication. Plusieurs systèmes de communication, tels que la télévision haute définition, le système mmWave, les large bande HF et les bandes ultra-large, disposent de canaux parcimonieux. Cette caractéristique peut être utilisée pour améliorer les performances de l'estimateur et réduire la taille de la séquence d'apprentissage, diminuant ainsi la puissance consommée et la bande passante. Cette thèse traite le problème de l'estimation du canal en explorant des méthodes qui exploitent sa parcimonie. L'étude de l'acquisition comprimée et de ses algorithmes a conduit à la proposition d'un nouvel algorithme appelé *Matching Pursuit based Least Square* (MPLS). L'utilisation de réseaux de neurones (NN) pour l'estimation de signaux parcimonieux a également été explorée. Les travaux ont été axés sur NN, inspirés d'algorithmes d'd'acquisition comprimée tels que *Learned Iterative Shrinkage-Thresholding Algorithm* (LISTA). Cela a abouti à deux approches qui améliorent les performances de LISTA ainsi qu'à un nouveau réseau de neurones adapté à l'estimation de signaux parcimonieux.

**Title :** Sparse Channels Estimation Applied in Software Defined Radio

**Keywords :** Compressive Sensing ; Neural Network ; Sparse Channel Estimation

**Abstract :** Communication channels are used to transmit information signals. However, these channels can cause several distortions on the signal to be transmitted, such as attenuation, multipath loss and Doppler shift, among others. For a better message recovery, the receiver can estimate the channel and bring more reliability to the communications systems. Several communications systems, for example high-definition television, mmWave system, wideband HF and ultra-wideband have sparse channels. This characteristic can be used to improve the performance of the estimator and reduce the size of the training sequence so decreasing the consumption power and bandwidth. This thesis handles the channel estimation problem by investigating methods that exploit the sparsity of the channel. The study of Compressive Sensing and its sparse recovery algorithms led to the proposition of a new algorithm called Matching Pursuit based on Least Square (MPLS). The use of neural networks (NN) to sparse signals estimation was also explored. The work focused on NN inspired by sparse recovery algorithms such as Learned Iterative Shrinkage-Thresholding Algorithm (LISTA). This resulted in two approaches that improve LISTA performance as well as to a new neural network suitable to estimate sparse signals.

**Institut Polytechnique de Paris**
91120 Palaiseau, France