TECHNICAL UNIVERSITY OF CRETE
ELECTRONIC AND COMPUTER ENGINEERING DEPARTMENT

Maximally Sparse Convex Channel Estimation and Equalization

by

George Lourakis

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DIPLOMA DEGREE OF

ELECTRONIC AND COMPUTER ENGINEERING

September 2014

THESIS COMMITTEE

Professor Athanasios P. Liavas, *Thesis Supervisor*
Professor Vassilis Digalakis
Associate Professor Aggelos Bletsas

Abstract

Sparse multipath channels are wireless links commonly found in communication systems such as High Frequency radio channels, horizontal and vertical underwater acoustic channels and terrestrial broadcasting channels for High Definition Television. Their impulse responses are characterized by a few significant terms that are widely separated in time. With high speed transmission, the length of a sampled sparse channel can reach hundreds of symbol interval. Thus, the amount of Intersymbol Interference (ISI) at the receiver is very high. Consequently, the presence of an ISI mitigating structure at the receiver, such as the Decision Feedback Equalizer (DFE), is essential. Due to the sparse impulse responses of these channels, traditional estimation techniques such as Least Squares (LS) result in overparameterization and thus poor performance of the estimator. Also, classical equalizers become too complex for tackling these channels. The problem of estimating and equalizing sparse multipath channels is considered in this thesis. We formulate the sparse channel estimation and the computation of the sparse DFE filters as sparse approximation problems. A usual approach in sparse approximation problems is regularization with an l_1 norm penalty term and usage of convex optimization techniques in order to acquire a solution [9, 10]. Other sparsity promoting penalty functions are available, but the l_1 norm has the advantage to be a convex function, making the l_1 norm regularized approximation problem a convex one. When a problem is formulated as a convex optimization problem, it can be solved by very fast, efficient and reliable algorithms. In order to achieve sparser solutions and still gain from the benefits of the convex optimization theory, the Maximally Sparse Convex (MSC) algorithm [1] utilizes a *non-convex* regularization term, that promotes sparsity more strongly than the l_1 norm, but chosen such that the total cost function remains convex. Details of the MSC algorithm are presented in chapter 2. Also, in chapter 2 some basic concepts of the optimization theory are recapitulated. In chapters 3 and 4, we study the application of the MSC algorithm in the sparse channel estimation and the sparse channel equalization, respectively. Finally, the conclusions of this thesis are presented in chapter 5.

Acknowledgements

I would like to thank my family for their support and encouragement. I would also like to thank my supervisor, Professor Athanasios Liavas, for his guidance throughout this work.

Table of Contents

List of Figures

List of Abbreviations

BER	Bit Error Rate
DFE	Decision Feedback Equalizer
FBF	Feedback Filter
FFF	Feedforward Filter
FIR	Finite Impulse Response
i.i.d.	independent and identically distributed
ISI	Intesymbol Interference
KKT	Karush Kuhn Tucker
LS	Least Squares
MLSE	Maximum Likelihood Sequence Estimation
MM	Majorization Minimization
MMSE	Minimum Mean Squared Error
MSC	Maximally Sparse Convex
MSE	Mean Squared Error
NP	Non-deterministic Polynomial-time
SNR	Signal to Noise Ratio
ZF	Zero Forcing

Chapter 1

Introduction

In this chapter, we introduce the channel model that will be used throughout this work. We briefly explain the importance of the channel equalization and present the most common categories of equalizers. We also introduce the notion of sparsity and we discuss some of its impacts in communications.

1.1 Channel Model

1.1.1 Multipath propagation

In wireless communication systems, the receiver observes a superposition of attenuated and delayed versions of the original transmitted radio signal, called multipath signal components. Reflections and refractions from obstacles and buildings are the cause of the multipath propagation phenomenon. Each path may be caused by a single reflector or by multiple reflectors clustered together with similar delays. The multipath effect is demonstrated in Figure 1.1.

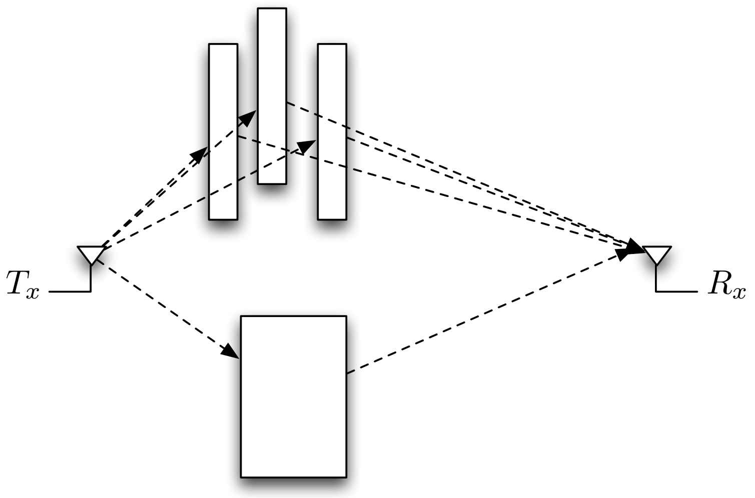


Figure 1.1: Multipath propagation.

The baseband equivalent of a multipath channel can be modelled as a linear time-varying filter, with impulse response

$$c(t,\tau) = \sum_{n=0}^{L(t)} c_n(t)\delta(\tau - \tau_n(t)), \tag{1.1}$$

where $L(t)$ is the number of channel paths, $c_n(t)$ is the complex gain of the n-th path at time t, and $\tau_n(t)$ is the delay of the n-th path at time t.

1.1.2 Multipath Fading

The signal attenuation and distortion due to multipath propagation is called multipath fading. The type of fading experienced by a signal propagating through a mobile radio channel depends on the nature of the transmitted signal with respect to the characteristics of the channel.

The coherence time of a channel is a measure of how quickly the channel response decorrelates. When the symbol period, T, is small compared to the coherence time, the fading is termed as slow fading. When the symbol period is comparable to the coherence time of the channel, the fading is termed fast. In the slow fading case, the channel may be assumed to be time invariant over several symbol periods. Thus, $L(t)$, $\tau_n(t)$ and $c_n(t)$ from (1.1) do not depend on time t, and the multipath channel becomes a linear time invariant filter with impulse response

$$c(\tau) = \sum_{n=0}^{L} c_n\delta(\tau - \tau_n). \tag{1.2}$$

For the rest of this work, we consider only the time invariant case.

Another classification of the fading process depends on the relationship between the delay spread of the channel, T_d, which is the difference in propagation time between the longest and shortest path, counting only the paths with significant energy, and the symbol period. When the delay spread is much smaller than the symbol period, the fading is classified as flat, and when it is not, it is termed as frequency selective fading.

The delay spread parameter is used to characterize the channel in the time domain. In the frequency domain the channel is characterized by the coherence bandwidth, B_c, which is the range of frequencies over which the signal strength remains more or less unchanged.

Flat Fading Channels

If the mobile radio channel has a constant gain and linear phase response over a bandwidth which is greater than the bandwidth of the transmitted signal, W, then the received signal will undergo frequency flat fading or simply, flat fading. This type of fading is historically the most common type of fading described in the technical literature. Flat fading channels are also known as amplitude varying channels and are sometimes referred to as narrowband channels, since the bandwidth of the applied signal is narrow as compared to the channel flat fading bandwidth or the Coherence bandwidth, B_c. To summarize, a signal undergoes flat fading if

$$W \ll B_c,$$

and

$$T \gg T_d.$$

The flat fading channel consists of one tap, thus, it can be viewed as a multiplicative channel.

Frequency Selective Channels

If the bandwidth of the transmitted signal is greater than the coherence bandwidth of the wireless channel, then it undergoes frequency selective fading. In such cases, the multipath delay spread is greater than the symbol interval. Consequently, the received signal contains multiple versions of the transmitted waveform which are attenuated and delayed in time and hence the received signal is distorted.

This kind of distortion is called Intersymbol Interference (ISI) and, roughly speaking, has similar effect as additive noise, thus, it makes the communication less reliable.

In the frequency domain, it is observed that different components have different gains than the others. Frequency selective fading channels are also called wideband channels since the symbol bandwidth is greater than the coherence bandwidth of the channel.

Thus, a channel undergoes frequency selective fading if

$$W \gtrapprox B_c,$$

and

$$T \lessapprox T_d.$$

Contrary to the frequency flat case where the channel consists of one tap, the frequency selective channel consists of multiple taps (resolvable multipaths). The multipath components are resolvable if they are separated in delay by T.

In this work, only frequency selective channels are studied.

1.1.3 Discrete-Time Channel Model

Most of the material in this subsection is from [12]. In digital communication systems, the transmitted signal consists of discrete symbols that are sampled at the symbol rate T and pulse shaped with the transmit filter $g_T(t)$. Both the transmit filter $g_T(t)$ and the receive filter $g_R(t)$, that will be mentioned later, are Square Root Raised Cosine filters. This pulse shaping operation at the transmitter is usually referred to as digital-to-analog conversion. Hence, the baseband transmitted signal can be written as

$$v(t) = \sum_k u_k g_T(t - kT),$$

where u_k is the transmit symbol sequence.

The transmitted signal is then convolved with the physical channel $c(t)$ and corrupted by the additive noise $n(t)$.

A basic and generally accepted model for thermal noise in communication channels is the set of the following assumptions. The noise is additive and statistically independent of the input signal. The noise is white, i.e, the power spectral density is flat, so the autocorrelation of the noise in time domain is zero for any non-zero time offset. The noise samples have a Gaussian distribution. This noise model, which is called Additive White Gaussian noise, is very efficient, even though the noise in reality is more complex.

Finally the received signal is filtered with the receive filter $g_R(t)$ and then sampled at the symbol rate T. The discretization operation at the receiver is normally referred to as analog-to-digital conversion.

The continuous-time communication system described above, is illustrate in Figure 1.2.

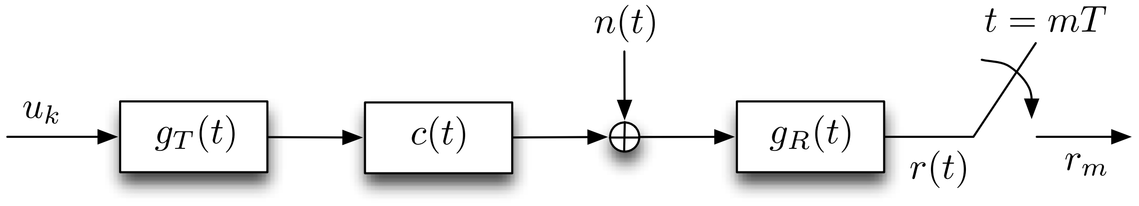


Figure 1.2: Continuous-time channel model.

We can write the input-output relation as

$$\begin{aligned} r(t) &= (v(t) * c(t) + n(t)) * g_R(t) \\ &= v(t) * c(t) * g_R(t) + g_R(t) * n(t) \\ &= u(t) * g_T(t) * c(t) * g_R(t) + n'(t), \end{aligned}$$

where $n'(t) = n(t) * g_R(t)$, which is also White Gaussian noise.

If we denote the overall channel impulse response $h(t) = g_T(t) * c(t) * g_R(t)$, then

$$\begin{aligned} r(t) &= u(t) * h(t) + n'(t) \\ &= \sum_k u_k h(t - nk) + n'(t). \end{aligned}$$

By taking samples $r_m = r(mT)$ we get

$$\begin{aligned} r_m &= r(mT) \\ &= r(t)\Big|_{t=mT} \\ &= \sum_k u_k h(t - kT)\Bigg|_{t=mT} + n'(t)\Big|_{t=mt} \\ &= \sum_k u_k h(mT - kT) + n'(mT). \end{aligned}$$

By setting $h_m = h(mT)$ and $n'_m = n'(mT)$, we have

$$r_m = \sum_k u_k h_{m-k} + n'_m. \tag{1.3}$$

Equation (1.3) describes the discrete-time baseband equivalent channel output as a convolution of the input symbol sequence u_k with the sampled overall impulse response h_k. This is a very useful expression, since it provides a general description of the input-output relation of the communication system, regardless the implementation methods, such as the transmit and receive filters or the physical channel.

In practice, h_k can be truncated to some finite length N. If we assume causality of $g_T(t)$, $g_R(t)$, and $c(t)$, then $h_k = 0$ holds for $k < 0$, and if N is chosen large enough $h_k \approx 0$ holds also for $l \geq N$. Therefore, (1.3) can be rewritten as:

$$r_m = \sum_{k=0}^{N-1} u_k h_{m-k} + n'_m. \tag{1.4}$$

Thus, the output of a communication system can be expressed as the output of

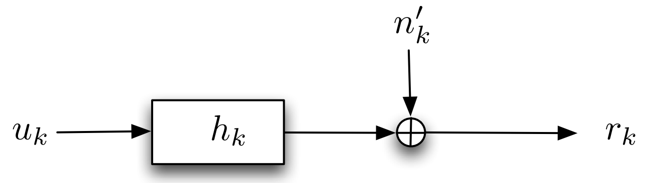


Figure 1.3: Discrete-time channel model.

a linear time invariant discrete-time system with input the symbol sequence of the communication system, as shown in Figure 1.3. This is the channel model we will use throughout this work.

1.2 Channel Equalization

The ability to transmit vast quantities of data through a reliable, high speed connection is very important. One way to achieve a high data rate is to simply increase the transmission speed. However, doing so inevitably increases the sampled channel length, thereby increasing the amount of ISI.

As mentioned before, ISI acts like additive noise, degrading severely the communication system. Thus, the utilization of techniques which cancel the effect of ISI are necessary. The techniques that aim in cancelling the ISI at the receiver of a communication system are called channel equalization.

1.2.1 Optimal receiver

In the receiver, Maximum Likelihood Sequence Estimation (MLSE) using the Viterbi algorithm on the received signal may be used to optimally reduce the effects of ISI [14].

The MLSE for a channel with ISI has a computational complexity that grows exponentially with the length of the channel time dispersion. If the size of the symbols alphabet is M and the number of interfering symbols contributing to ISI is L, the Viterbi algorithm computes M^{L+1} metrics for each received symbol. In many channels of practical interest, such a large computational complexity is prohibitively expensive to implement.

1.2.2 Sub-optimal Equalizers

The computational complexity of the MLSE leads to an interest in seeking sub-optimal equalizers. These equalizers can be divided into categories [15].

Linear or Non-Linear Equalizers

Two broad categories of the equalization techniques are the linear and the non-linear. The linear techniques are generally the simplest to implement and to understand conceptually. A linear equalizer is a filter that equalizes an ISI channel by inverting its frequency response. It is most effective when applied on mild ISI channels whose frequency responses are relatively flat.

However, linear equalization techniques typically suffer from more noise enhancement than non-linear equalizers, and are therefore not used in most wireless applications. Among non-linear equalization techniques, Decision Feedback Equalization (DFE) is the most common, since it is fairly simple to implement and generally performs well. The DFE consists of a Feedforward Filter (FFF) with the received sequence as input, followed by a Feedback Filter (FBF) with the previously detected sequence as input. Effectively, the DFE determines the ISI contribution from the already detected symbols by passing them through the FBF. The resulting ISI is then subtracted from the incoming symbols. The FBF of the DFE does not suffer from noise enhancement because it estimates the channel frequency response rather than its inverse. For channels with deep spectral nulls, DFEs generally perform much better than linear equalizers. However, on channels with low Signal to Noise Ratio (SNR), the DFE suffers from error propagation when symbols are decoded in error, leading to poor performance.

Zero-Forcing or MMSE Equalizers

The two most common criteria for selecting the coefficient of a linear or a non-linear equalizer are the total elimination of the ISI, and the minimization of the Mean Squared Error (MSE). An equalizer implemented using the first strategy is called Zero Forcing (ZF) equalizer. A serious problem with the ZF equalizer is the noise enhancement, which can result in infinite noise power spectral densities after the equalizer. The noise is enhanced at frequencies where the channel has a high attenuation. An equalizer implemented using the second strategy is called Minimum Mean Squared Error (MMSE) equalizer. The MMSE equalizer balances a reduction in ISI with noise enhancement. The MMSE equalizer always performs as well as, or better than, the ZF equalizer.

1.3 Sparsity

1.3.1 Definition

The term sparsity refers to a measurable property of a vector. It means that the number of the non-zero elements of the vector is very small in comparison with its dimension.

There are a lot of advantages working with sparse vectors. For example calculations involving multiplying a vector by a matrix take less time to compute in general if the vector is sparse. Also sparse vectors require less space when being stored on a computer as only the position and value of the entries need to be recorded.

A very simple and intuitive measure of sparsity of a vector $\mathbf{x}$ simply involves the number of non-zero entries in $\mathbf{x}$. The vector is sparse if there are few non-zeros among the possible entries in $\mathbf{x}$. It will be convenient to introduce the l_0 quasi-norm

$$||\mathbf{x}||_0 = \# \{i : x_i \neq 0\} . \tag{1.5}$$

Thus if $||\mathbf{x}||_0$ is considerably less than the vector length, $\mathbf{x}$ is sparse.

1.3.2 Sparse Multipath Channels

Channel measurement results suggest that multipath components tend to be distributed in clusters rather than uniformly over the channel delay spread. These clusters of paths physically correspond to large-scale objects in the scattering environment, such as buildings and hills in an outdoor propagation environment, while multipath components within a cluster arise as a result of scattering from small-scale structures of the corresponding large-scale reflector, such as windows of a building or trees on a hill. Based on the interarrival times between different multipath clusters within the delay spread, wireless channels can be categorized as either rich or sparse.

A sparse multipath channel is characterized by an impulse response that only comprises a few significant multipath terms. However, the multipath terms are widely separated in time, thereby creating a large delay spread. With high speed transmission, 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.

Some examples of sparse multipath channels are the High Frequency radio channels, where the large time differences between the multipath terms are caused by reflections off the ionosphere, horizontal and vertical underwater acoustic channels, where the long delays between the multipath terms are due to reflections off the sea surface or sea floor, terrestrial broadcasting channels for High Definition Television systems, cellular land mobile radio channels in hilly environments and aeronautical channels [7].

An example of an impulse response of a sparse multipath channel of 100 taps length and significantly less non-zero taps is illustrated in Figure 1.4.

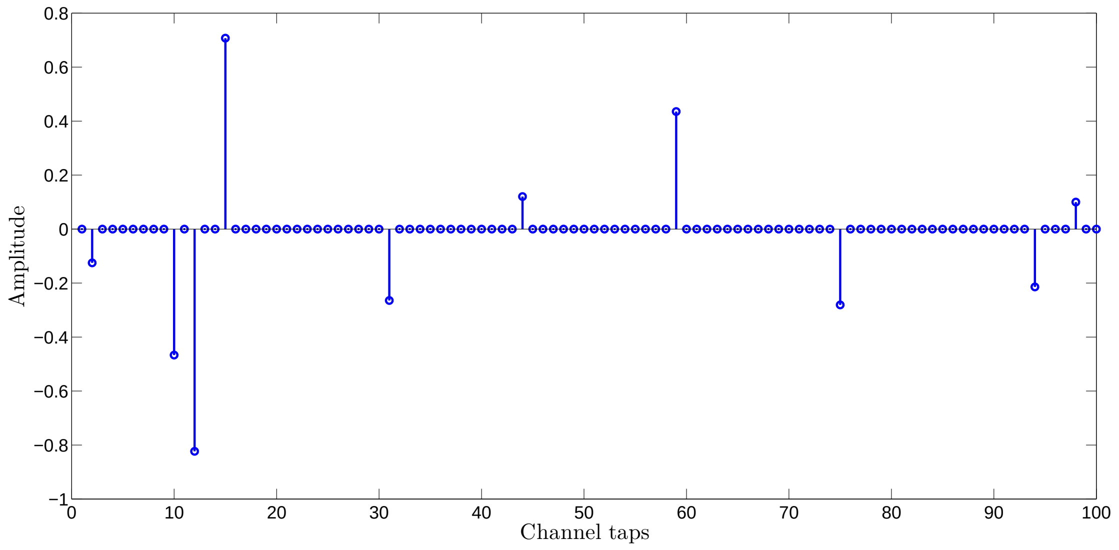


Figure 1.4: Sparse Multipath Channel.

1.3.3 Sparse Channel Estimation

An exact estimate of the sparse multipath channel will greatly improve the performance of the equalization. Typically, this is accomplished by probing the channel with a known training sequence and processing the channel output, using for example a conventional Least Squares (LS) based channel estimation scheme.

While appropriate for rich channels, LS estimation ignores the structure of a sparse multipath channel, which leads to poor performance, since estimation effort is directed towards estimating all the channel coefficients, many of which might be zero.

Thus, other channel estimation techniques, which exploit the sparsity of the sparse multipath channel, should be used. The performance of the LS estimation and the sparse channel estimation techniques will be discussed in chapter 3 with more details.

1.3.4 Sparse Equalizers

In applications where sparse multipath channels are considered, very long equalizers have to be employed at the receiver to mitigate the resulting severe ISI. The complexity of computing and implementing FIR equalizers increases with the number of taps. Thus, the complexity of the equalization of a sparse multipath channel may become prohibitive.

In systems where the complexity is dominated by arithmetic operations, the number of non-zero coefficients in the impulse response may be a more appropriate metric to consider instead, and computational savings are realized by omitting arithmetic operations associated with zero-valued coefficients. This leads to a demand for long equalizers with fewer non-zero coefficients (sparse equalizers), to reduce complexity at the expense of some performance loss. Thus, the sparse equalizer is better structured to handle sparse multipath channels with large delay spreads because the computational complexity of the filtering operation is no longer proportional to the equalizer span.

In addition to complexity reduction, under non-ideal channel estimation, the sparse structure can improve the equalizer's performance because it reduces the noise accumulation caused by the non-ideal computation of the equalizer weights.

Techniques for computing sparse DFE filters will be studied in chapter 4.

Chapter 2

Convex Optimization

Convex optimization refers to the minimization of a convex objective function subject to convex constraints [13]. Convex optimization has found wide application in areas such as automatic control systems, estimation and signal processing, communications and networks, electronic circuit design, data analysis and modelling, statistics, and finance. Many of the most commonly addressed optimization problems are convex. Or even if they are not convex, some times, there is a way to reformulate them into convex form. There are great advantages to recognizing or formulating a problem as a convex optimization problem. The most basic advantage is that, due to the convex nature of the feasible set of the problem, any local optimum is also the global optimum. Algorithms written to solve convex optimization problems take advantage of such properties, and are faster, more efficient and very reliable. As a result, very large problems, with even thousands of variables and constraints, are solvable in reasonably small time. Thus, these algorithms can be embedded in a computer-aided design or analysis tool, or even a real-time reactive or automatic control system [8].

2.1 Basic Optimization Concepts

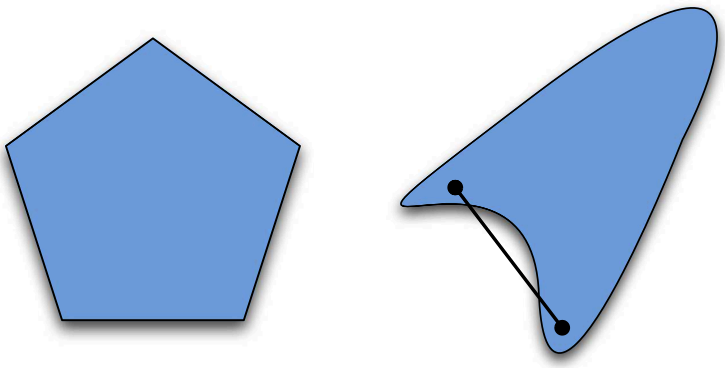
In order to recognize and solve convex optimization problems in engineering applications, one must first be familiar with the basic concepts of convex optimization theory. This section provides a concise review of these optimization concepts. Most of the material in this section is from [13].

2.1.1 Convex Set

A set C is convex if the line segment between any two points of C lies in C. That is, if for any $x_1, x_2 \in C$ and any θ with $0 \leq \theta \leq 1$, we have

$$\theta x_1 + (1-\theta)x_2 \in C.$$

Roughly speaking, a set is convex if every point in the set can be seen by every other point, along an unobstructed straight path between them, where unobstructed means lying in the set, as shown in Figure 2.1.



(a) Convex set, (b) Non-convex set.

Figure 2.1: Convex and non-convex sets.

2.1.2 Convex Function

A function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if the $\mathbf{dom} f$ is a convex set and for all $\mathbf{x}, \mathbf{y} \in \mathbf{dom} f$, and $\theta \in \mathbb{R}$, with $0 \leq \theta \leq 1$, we have

$$f(\theta \mathbf{x} + (1-\theta)\mathbf{y}) \leq \theta f(\mathbf{x}) + (1-\theta) f(\mathbf{y}). \tag{2.1}$$

Geometrically, this inequality means that the line segment between $(\mathbf{x}, f(\mathbf{x}))$ and $(\mathbf{y}, f(\mathbf{y}))$ lies above the graph of f. The visualization of this geometrical interpretation, for the simple case where $f : \mathbb{R} \to \mathbb{R}$ and $x, y \in \mathbb{R}$, is illustrated in Figure 2.2. A function f is strictly convex if strict inequality holds in (2.1) whenever $\mathbf{x} \neq \mathbf{y}$ and $0 < \theta < 1$. We say f is concave if $-f$ is convex, and strictly concave if $-f$ is strictly convex.

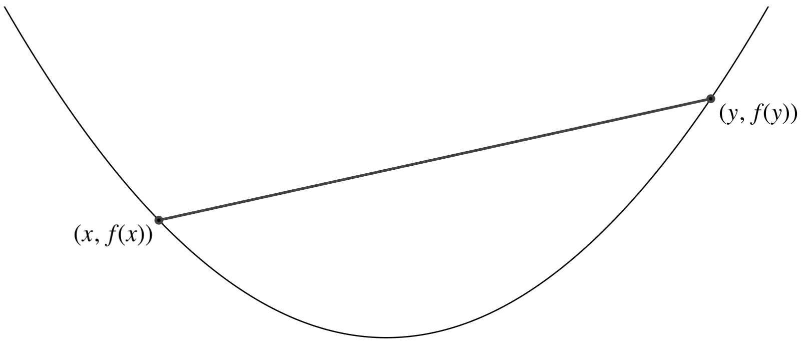


Figure 2.2: Convex function.

2.1.3 Gradient and Subdifferential

If $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, then the function $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ defined as

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial f}{\partial x_n}(\mathbf{x}) \end{bmatrix},$$

is called gradient of f at $\mathbf{x}$.

An important property of a convex and differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is that, for every $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, we have

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x}).$$

A geometrical interpretation of this property for the simple case where $f : \mathbb{R} \to \mathbb{R}$ and $x, y \in \mathbb{R}$, is illustrated in Figure 2.3.

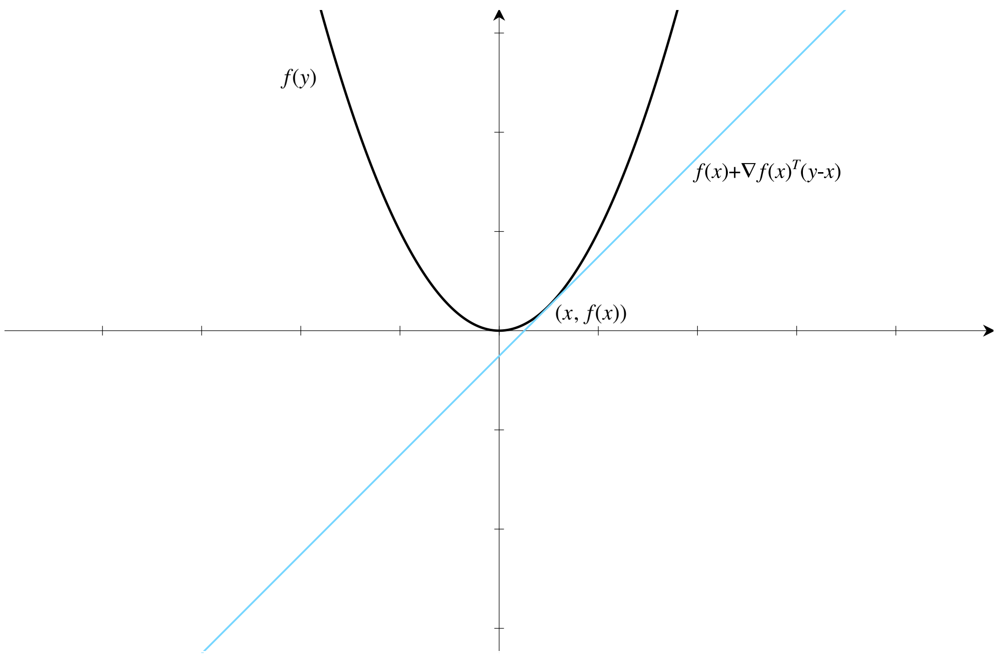


Figure 2.3: Important property of gradient.

When the function of interest is not differentiable everywhere, its gradient can not be computed at the non-smooth points. At these points, we use the subdifferential of the function.

We define $\mathbf{g}$ as a subgradient of a function f, at $\mathbf{x}$, if

$$f(\mathbf{y}) \geq f(\mathbf{x}) + \mathbf{g}^T(\mathbf{y} - \mathbf{x}), \quad \forall \mathbf{y} \in \mathbf{dom} f.$$

The set of all subgradients of f at $\mathbf{x}$ is called the subdifferential of f at $\mathbf{x}$ and is denoted as

$$\partial f(\mathbf{x}) = \{\mathbf{g}|\mathbf{g}^T(\mathbf{y}-\mathbf{x}) \leq f(\mathbf{y}) - f(\mathbf{x}), \quad \forall \mathbf{y} \in \mathbf{dom} f\}.$$

A geometrical representation of the subdifferential for the simple case where $f : \mathbb{R} \to \mathbb{R}$ and $x, y \in \mathbb{R}$, is illustrated in Figure 2.4.

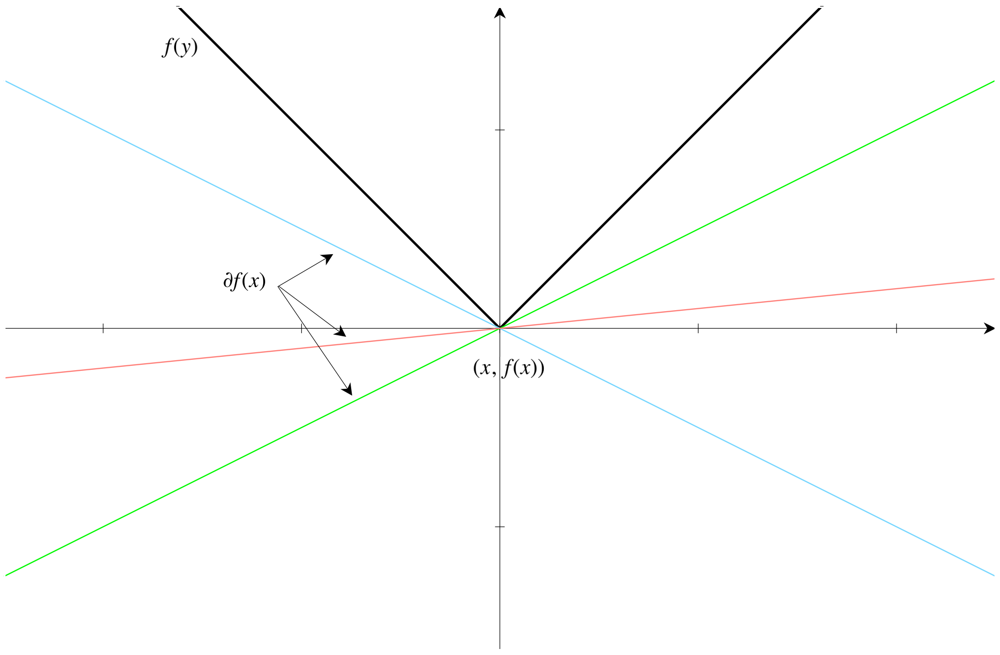


Figure 2.4: Subdifferential of $f(x)$.

2.1.4 Convex Optimization Problem

A convex optimization problem is one of the form

$$\begin{aligned} \text{minimize} \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & f_i(\mathbf{x}) \leq 0, \qquad i = 1, \ldots, m \\ & h_i(\mathbf{x}) = 0, \qquad i = 1, \ldots, p, \end{aligned} \tag{2.2}$$

where $f_0, \ldots, f_m$ are convex functions and $h_1, \ldots, h_p$ are affine functions, that is, $h_i(\mathbf{x}) = \mathbf{a}_i^T\mathbf{x} - b_i$. We call $\mathbf{x} \in \mathbb{R}^n$ the optimization variable and the function $f_0 : \mathbb{R}^n \to \mathbb{R}$ the objective or cost function. The inequalities $f_i(\mathbf{x}) \leq 0$ are called the inequality constraints, and the corresponding functions $f_i : \mathbb{R}^n \to \mathbb{R}$ are called the inequality constraint functions. The equations $h_i(\mathbf{x}) = 0$ are called the equality constraints, and the functions $h_i : \mathbb{R}^n \to \mathbb{R}$ are the equality constraint functions. If there are no constraints, we say the problem (2.2) is unconstrained.

The set of points for which the objective and all constraint functions are defined is called the domain $\mathcal{D}$ of the convex optimization problem (2.2). A point $\mathbf{x} \in \mathcal{D}$ is feasible if it satisfies the constraints $f_i(\mathbf{x}) \leq 0,\ i = 1, \ldots, m$, and $\mathbf{a}_i^T\mathbf{x} = b_i,\ i = 1, \ldots, p$. The problem (2.2) is said to be feasible if there exists at least one feasible point, and infeasible otherwise. The set of all feasible points is called the feasible set or the constraint set. The feasible set of a convex optimization problem is convex, since it is the intersection of the domain $\cap_{i=0}^{m}\mathbf{dom} f_i$, which is a convex set, with m sublevel sets and p hyperplanes. Thus, in a convex optimization problem, we minimize a convex objective function over a convex set.

2.1.5 Lagrangian

In order to define the Langrangian function of a convex optimization problem, we take the constraints in (2.2) into account by augmenting the objective function with a weighted sum of the constraint functions. We define the Lagrangian $L : \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ associated with the problem (2.2) as

$$L(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{v}) = f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} v_i h_i(\mathbf{x}),$$

with $\mathbf{dom} L = \mathcal{D} \times \mathbb{R}^m \times \mathbb{R}^p$. We refer to λ_i as the Lagrange multiplier associated with the i-th inequality constraint $f_i(\mathbf{x}) \leq 0$. Similarly, we refer to v_i as the Lagrange multiplier associated with the i-th equality constraint $h_i(\mathbf{x}) = 0$. The vectors $\boldsymbol{\lambda}$ and $\mathbf{v}$ are called the dual variables or Lagrange multiplier vectors associated with the problem (2.2).

2.1.6 Lagrange Dual Function

The Lagrange Dual Function (or just dual function) $g : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ is defined as the minimum value of the Lagrangian over $\mathbf{x}$: for $\boldsymbol{\lambda} \in \mathbb{R}^m$, $\mathbf{v} \in \mathbb{R}^p$

$$\begin{aligned} g(\boldsymbol{\lambda}, \mathbf{v}) &= \inf_{\mathbf{x} \in \mathbf{dom}\mathcal{D}} L(\mathbf{x}, \boldsymbol{\lambda}, \mathbf{v}) \\ &= \inf_{\mathbf{x} \in \mathbf{dom}\mathcal{D}} \left(f_0(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i f_i(\mathbf{x}) + \sum_{i=1}^{p} v_i h_i(\mathbf{x}) \right). \end{aligned}$$

When the Lagrangian is unbounded below in $\mathbf{x}$, the dual function takes on the value $-\infty$. Since the dual function is the pointwise infimum of a family of affine functions of $(\boldsymbol{\lambda}, \mathbf{v})$, it is concave, even when the problem (2.2) is not convex.

An important property of the dual function is that it yields lower bounds on the optimal value p_* of the problem (2.2): For any $\boldsymbol{\lambda} \geq 0$ and any $\mathbf{v}$, we have

$$g(\boldsymbol{\lambda}, \mathbf{v}) \leq p_*.$$

2.1.7 Dual Problem

The dual function provides a lower bound that depends on parameters $\boldsymbol{\lambda}, \mathbf{v}$. The best lower bound that can be obtained from the dual function is computed by the optimization problem

$$\begin{aligned} \text{maximize} \quad & g(\boldsymbol{\lambda}, \mathbf{v}) \\ \text{subject to} \quad & \boldsymbol{\lambda} \geq \mathbf{0}. \end{aligned} \tag{2.3}$$

This problem is called the Lagrange dual problem associated with the problem (2.2). In this context, the original problem (2.2) is sometimes called the primal problem. The term dual feasible is used to describe a pair $(\boldsymbol{\lambda}, \mathbf{v})$ with $\boldsymbol{\lambda} \geq \mathbf{0}$ and $g(\boldsymbol{\lambda}, \mathbf{v}) \geq -\infty$. We refer to $(\boldsymbol{\lambda}_*, \mathbf{v}_*)$ as dual optimal or optimal Lagrange multipliers if they are optimal for the problem (2.3).

The Lagrange dual problem (2.3) is a convex optimization problem, since the objective to be maximized is concave and the constraint is convex. This is the case whether or not the problem (2.2) is convex.

The optimal value of the Lagrange dual problem, which we denote d_*, is, by definition, the best lower bound on p_* that can be obtained from the Lagrange dual function. In particular, we have the simple but important inequality $d_* \leq p_*$, which holds even if the original problem is not convex. This property is called weak duality.

We refer to the difference $p_* - d_*$ as the optimal duality gap of the original problem, since it gives the gap between the optimal value of the primal problem and the best lower bound on it that can be obtained from the Lagrange dual function. The optimal duality gap is always nonnegative.

If the equality $d_* = p_*$ holds, i.e., the optimal duality gap is zero, then we say that strong duality holds. This means that the best bound that can be obtained from the Lagrange dual function is tight. Strong duality does not, in general, hold. But if the primal problem (2.2) is convex , we usually (but not always) have strong duality.

2.1.8 Optimality Conditions

When a convex optimization problem is unconstrained and the objective function f_0 is differentiable for all $\mathbf{x} \in \mathbf{dom} f_0$, then, $\mathbf{x}_0$ is optimal if and only if

$$\nabla f_0(\mathbf{x}_0) = \mathbf{0}.$$

When a convex optimization problem is unconstrained, but the objective function f_0 is not differentiable for all $\mathbf{x} \in \mathbf{dom} f_0$, then, $\mathbf{x}_0$ is optimal if and only if

$$\mathbf{0} \in \partial f_0(\mathbf{x_0}).$$

For any optimization problem with differentiable objective and constraint functions for which strong duality obtains, any pair of primal and dual optimal points, $\mathbf{x}_*$ and $(\boldsymbol{\lambda}_*,\ \mathbf{v}_*)$ must satisfy the Karush Kuhn Tucker (KKT) conditions

$$\begin{aligned} \nabla f_0(\mathbf{x}_*) + \sum_{i=1}^{m} \lambda_{*,i} \nabla f_i(\mathbf{x}_*) + \mathbf{A}^T \mathbf{v}_* &= \mathbf{0}, \\ \mathbf{A}\mathbf{x}_* &= \mathbf{b}, \\ f_i(\mathbf{x}_*) &\leq 0, \qquad i = 1, \dots, m, \\ \boldsymbol{\lambda}_* &\geq \mathbf{0}, \\ \lambda_{*,i} f_i(\mathbf{x}_*) &= 0, \qquad i = 1, \dots, m. \end{aligned}$$

When the primal problem is convex, as in (2.2), the KKT conditions are also sufficient for the points to be primal and dual optimal. In other words, if f_i are convex and h_i are affine, and $\tilde{\mathbf{x}}$, $\tilde{\boldsymbol{\lambda}}$, $\tilde{\mathbf{v}}$ are any points that satisfy the KKT conditions, then $\tilde{\mathbf{x}}$ and $(\tilde{\boldsymbol{\lambda}},\ \tilde{\mathbf{v}})$ are primal and dual optimal, with zero duality gap.

2.1.9 Majorization Minimization Method

The Majorization Minimization (MM) framework substitutes a difficult optimization problem with a simpler one. It works by finding a surrogate function that minorizes or majorizes the objective function. Optimizing the surrogate functions will drive the objective function upward or downward until a local optimum is reached. Instead of minimizing the objective function $F(\mathbf{x})$ directly, the MM method solves a sequence of simpler minimization problems

$$\mathbf{x}_{k+1} = \underset{\mathbf{x}}{\operatorname{argmin}}\ G_k(\mathbf{x}),$$

where k is the iteration counter. Each function $G_k(\mathbf{x})$ is a majorizer (upper bound) for $F(\mathbf{x})$ and coincides with $F(\mathbf{x})$ at $\mathbf{x} = \mathbf{x}_k$. That is

$$\begin{aligned} G_k(\mathbf{x}) &\geq F(\mathbf{x}), \quad \forall\, \mathbf{x}, \\ G_k(\mathbf{x_k}) &= F(\mathbf{x_k}). \end{aligned}$$

The MM procedure monotonically reduces the cost function value at each iteration. Under mild conditions, the sequence $\mathbf{x}_k$ converges to the minimizer of $F(\mathbf{x})$.

2.2 Regularization and Sparse Problems

We consider the system

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{w}, \tag{2.4}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a sparse signal, $\mathbf{y} \in \mathbb{R}^m$ is the observed signal, $\mathbf{H} \in \mathbb{R}^{m \times n}$ is a linear operator, and $\mathbf{w} \in \mathbb{R}^m$ is additive white Gaussian noise vector.

When the structure of $\mathbf{x}$ is unknown, an approximation of $\mathbf{x}$ is computed by the least squares optimization problem

$$\min_{\mathbf{x}} \quad ||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2.$$

On the other hand, if the structure of $\mathbf{x}$ is known, for example, if it has many small values or a sparse form, then the regularized approximation is very useful.

In the basic form of regularized approximation the goal is to find a vector $\mathbf{x}$ that is small, and also makes the residual $(\mathbf{y} - \mathbf{H}\mathbf{x})$ small. This is naturally described as a convex optimization problem with two objectives, $||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2$ and $||\mathbf{x}||$. A common scalarization method used to solve an optimization problem with two objectives is to minimize the weighted sum of the objectives. The two norms can be different. The first, used to measure the size of the residual, is on $\mathbb{R}^m$ and the second, used to measure the size of $\mathbf{x}$, is on $\mathbb{R}^n$, and is referred to as the penalty function.

The norm used as a penalty function will depend on our knowledge about the structure of $\mathbf{x}$. For example, when we want to obtain an approximation with small values, the well known l_2 norm is suitable to be used as a penalty function. In our case, where $\mathbf{x}$ is sparse, the natural approach towards the sparse regularized approximation problem is to use the l_0 quasi-norm, defined in (1.5), as the penalty function. Thus, the sparse regularized approximation problem is of the form

$$\min_{\mathbf{x}} \quad \frac{1}{2}||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2 + \lambda||\mathbf{x}||_0, \tag{2.5}$$

where $\lambda > 0$. The sparsity of the approximation depends on λ, with $\mathbf{x}$ getting more sparse, as λ increases.

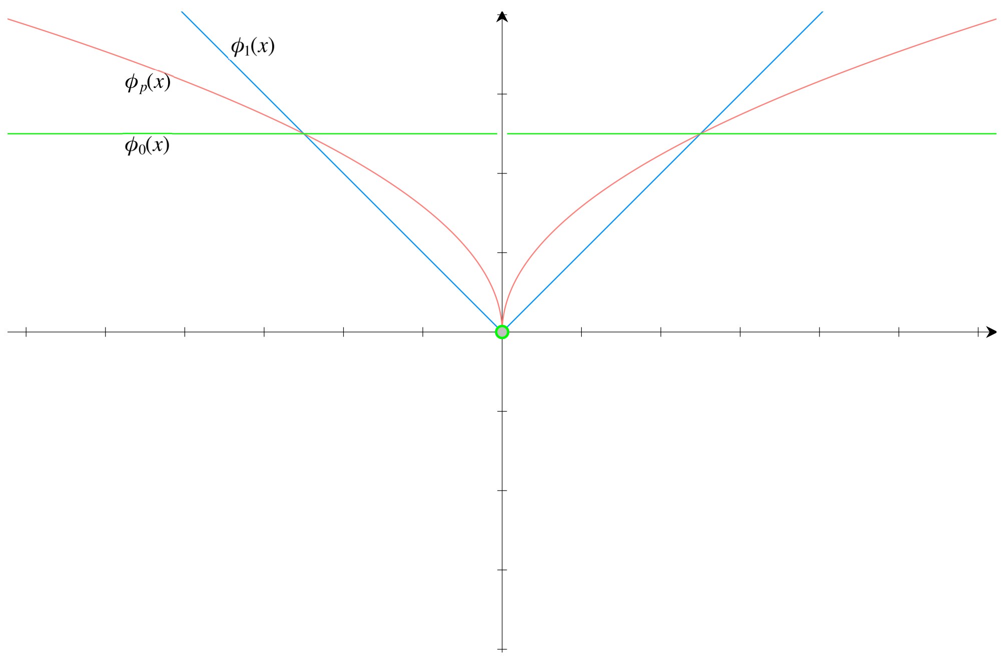


Figure 2.5: Penalty functions.

We can express the l_0 quasi-norm as

$$||\mathbf{x}||_0 = \sum_i \phi_0(x_i),$$

where $\phi_0(x)$ is illustrated in Figure 2.5, and is given by

$$\phi_0(x) = \begin{cases} 0, & \text{if } x = 0, \\ 1, & \text{if } x \neq 0. \end{cases}$$

Thus, the optimization problem (2.5) can be rewritten as

$$\min_{\mathbf{x}} \quad \frac{1}{2}||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2 + \lambda \sum_i \phi_0(x_i). \tag{2.6}$$

Unfortunately, the solution of (2.6) is very difficult to obtain. In order to obtain the solution, we have to solve exhaustively, for all the possible values of $||\mathbf{x}||_0$, the least squares problem $||\mathbf{y}-\mathbf{H}\mathbf{x}||_2^2$, making the problem Non-deterministic Polynomial-time (NP)-hard. The difficulty of this problem is rooted in the discrete and discontinuous nature of $\phi_0(x)$.

Searching for a candidate substitute for the l_0 quasi-norm in the sparse approximation problem, we examine the l_p quasi-norm with $0 < p < 1$, defined as

$$||\mathbf{x}||_p = \left(\sum_i |x_i|^p \right)^{1/p}.$$

Then, the sparse regularized approximation problem (2.5) becomes

$$\min_{\mathbf{x}} \quad \frac{1}{2}||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2 + \lambda ||\mathbf{x}||_p^p, \tag{2.7}$$

If we define

$$\phi_p(x) = |x|^p,$$

then, the optimization problem (2.7) can be rewritten as

$$\min_{\mathbf{x}} \quad \frac{1}{2}||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2 + \lambda \sum_i \phi_p(x_i). \tag{2.8}$$

As shown in Figure 2.5, $\phi_p(x)$ is a continuous function which is geometrically very close to $\phi_0(x)$. Unfortunately, $\phi_p(x)$ is a non-convex function, making the solution of (2.8) difficult. Thus, $\phi_p(x)$ is not a good candidate substitute of the $\phi_0(x)$ in the sparse approximation problem.

We can get a convex relaxation of (2.5) by replacing the l_0 quasi-norm with the l_1 norm, which is defined as

$$||\mathbf{x}||_1 = \sum_i |x_i|.$$

Then, the sparse regularized approximation problem (2.5) becomes

$$\min_{\mathbf{x}} \quad \frac{1}{2}||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2 + \lambda ||\mathbf{x}||_1. \tag{2.9}$$

If we define

$$\phi_1(x) = |x|, \tag{2.10}$$

then, the optimization problem (2.9) can be rewritten as

$$\min_{\mathbf{x}} \quad \frac{1}{2}||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2 + \lambda \sum_i \phi_1(x_i), \tag{2.11}$$

With a geometrical observation of Figure 2.5, we can say that, among convex functions, $\phi_1(x)$ is, in some sense, the closest to the $\phi_0(x)$. Thus, given the relative ease with which convex problems can be reliably solved, the l_1 norm is a basic tool in sparse signal processing.

2.3 Maximally Sparse Convex Optimization

Convex optimization with sparsity-promoting convex (l_1 norm) regularization is a standard approach for estimating sparse signals in noise. In order to promote sparsity more strongly than convex regularization, it is also standard practice to employ non-convex optimization. The Maximally Sparse Convex (MSC) approach [1] utilizes a non-convex regularization term chosen such that the total cost function, consisting of the sum of the residual and the regularization terms, is convex. Therefore, sparsity is more strongly promoted than in the standard convex formulation, but without sacrificing the attractive aspects of convex optimization, such as the unique minimum or the robust algorithms provided by convex optimization theory. Most of the material in this section is from [1].

In essence, MSC optimization attempts to solve sparse regularized least squares approximation problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad F(\mathbf{x}) = \frac{1}{2}||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2 + \sum_{i=0}^{n-1} \lambda_i \phi(x_i; a_i), \tag{2.12}$$

where $\lambda_i > 0$, for $i = 1, \dots, n$, and $\phi(x; a)$ is a non-convex sparsity promoting penalty function, with the parameters a_i selected so as to ensure convexity of the total cost function $F(\mathbf{x})$.

2.3.1 Sparsity Promoting Penalty Functions

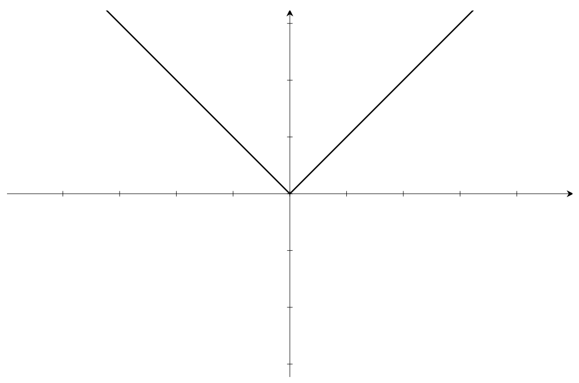
One penalty function suitable for the MSC approach is the logarithmic function defined as

$$\phi_{\log}(x) = \frac{1}{a}\log(1 + a|x|), \quad a > 0. \tag{2.13}$$

As a approaches zero, the logarithmic function approaches the l_1 norm, providing no significant advantage. As a increases, the logarithmic function becomes more and more non-convex, promoting sparser solutions.

In order to study non-convex sparsity promoting penalty functions, we will compare the non-convex logarithmic penalty function with the convex l_1 norm. Both penalty functions are illustrated in Figure 2.6

First, we will introduce the term threshold function. Each penalty function is related with one threshold function, which can give us some information about the


(a) $\phi_1(x)$

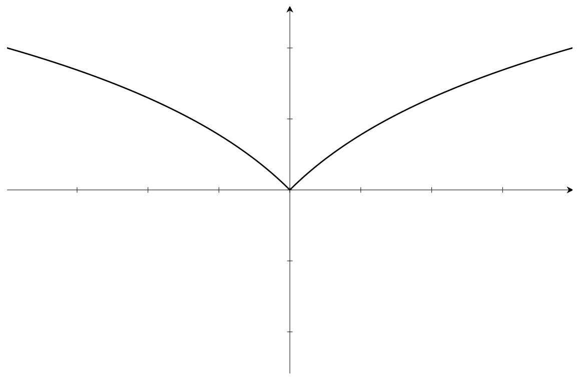

(b) $\phi_{log}(x)$

Figure 2.6: Penalty functions.

performance of the penalty function [1].

If $\mathbf{H}$ in (2.12) is the identity matrix, $F(\mathbf{x})$ is separable in scalars. Thus, if $x, y \in \mathbb{R}$ we have

$$F(x) = \frac{1}{2}(y - x)^2 + \lambda\phi(x),$$

and the minimization problem (2.12) becomes

$$\theta(y) = \underset{x\in\mathbb{R}}{\operatorname{argmin}}\left\{\frac{1}{2}(y - x)^2 + \lambda\phi(x)\right\}.$$

The function $\theta(y)$ is the threshold function and depends on the penalty function $\phi(x)$.

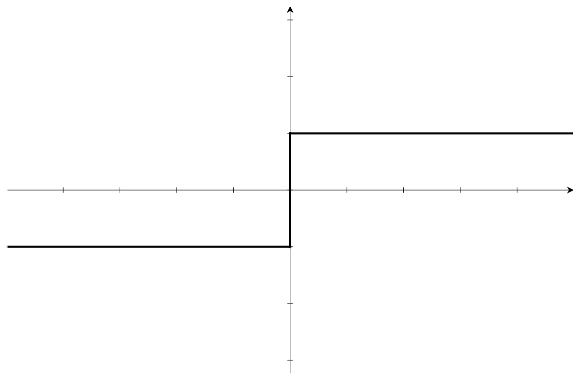
The threshold function of a penalty function is computed by minimizing $F(x)$, with respect to x. The derivative of $F(x)$ is given by

$$F'(x) = x - y + \lambda\phi'(x).$$

Setting the derivative of $F(x)$ equal to zero gives

$$y = x + \lambda\phi'(x). \tag{2.14}$$

The value x that minimizes $F(x)$ can be found by solving (2.14). First, the derivatives of the penalty functions must be found. It is obvious that both penalty functions are differentiable for all $x \in \mathbb{R}$, except $x = 0$. Thus, the subdifferential $\partial\phi(x)$ must be computed


(a) $\partial\phi_1(x)$

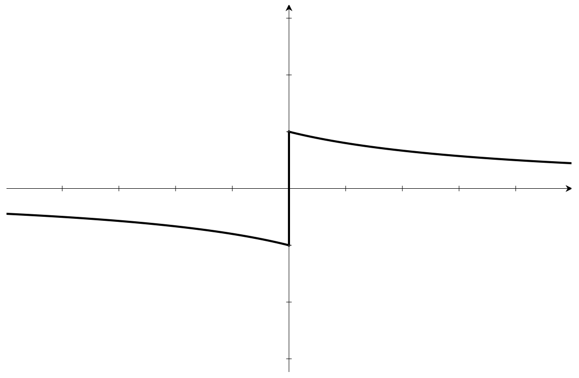

(b) $\partial\phi_{\log}(x)$

Figure 2.7: Subdifferentials of the penalty functions.

$$\partial\phi(x) = \begin{cases} \phi'(x), & \text{for } x \neq 0, \\ [\phi'(0^-), \phi'(0^+)], & \text{for } x = 0. \end{cases}$$

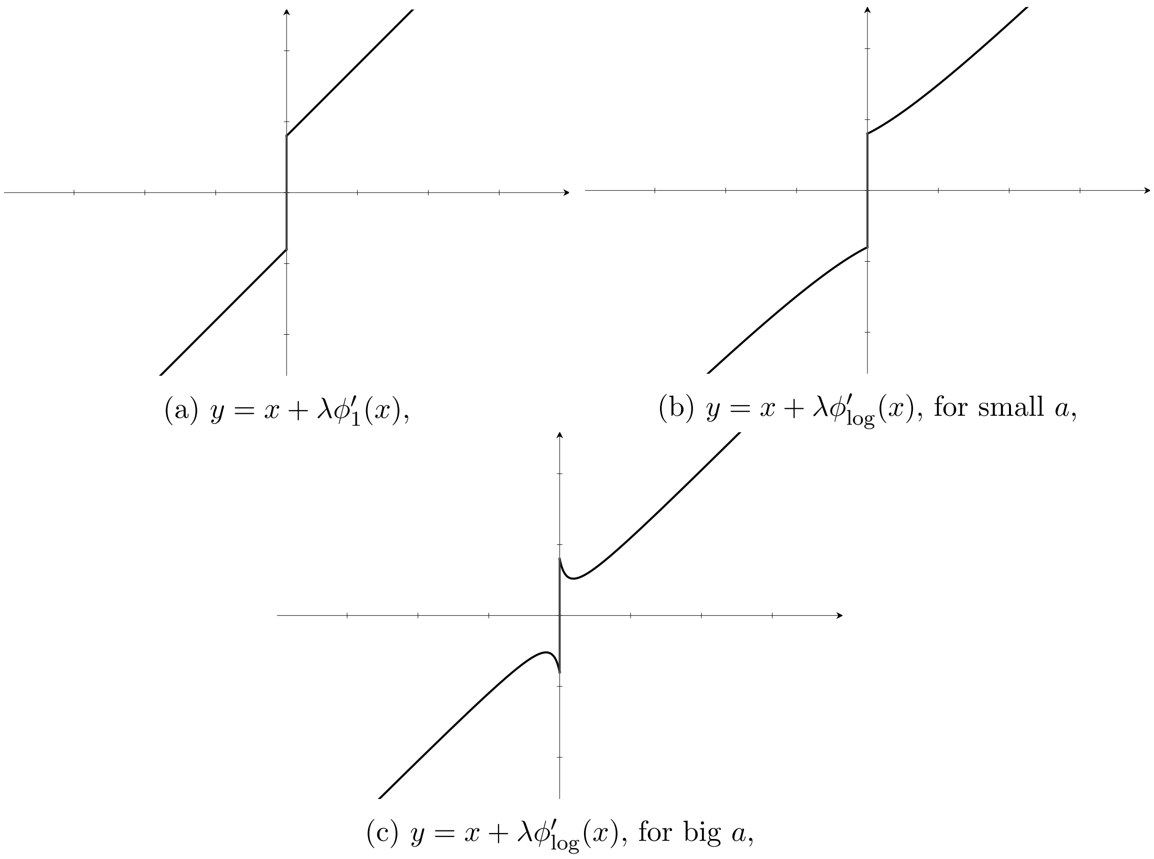
For each penalty function, we have

$$\partial\phi_1(x) = \begin{cases} 1, & \text{for } x > 0, \\ [-1, 1], & \text{for } x = 0, \\ -1, & \text{for } x < 0, \end{cases}$$

$$\partial\phi_{\log}(x) = \begin{cases} \dfrac{1}{1+a|x|}, & \text{for } x > 0, \\ [-1,1], & \text{for } x = 0, \\ -\dfrac{1}{1+a|x|}, & \text{for } x < 0. \end{cases}$$

The subdifferential of the penalty functions are illustrated in Figure 2.7.

Now, that we have obtained the $\phi'(x)$, it is easy to form the $y = x + \lambda\phi'(x)$. The case where we use the l_1 norm is illustrated in Figure 2.8a. For the logarithmic function case we take two values for a, one small and one big. These cases are illustrated in Figure 2.8b and Figure 2.8b respectively. We observe that for a big value for a, $y = x + \lambda\phi'(x)$ is not a strictly increasing function of x.


(a) $y = x + \lambda\phi_1'(x)$,

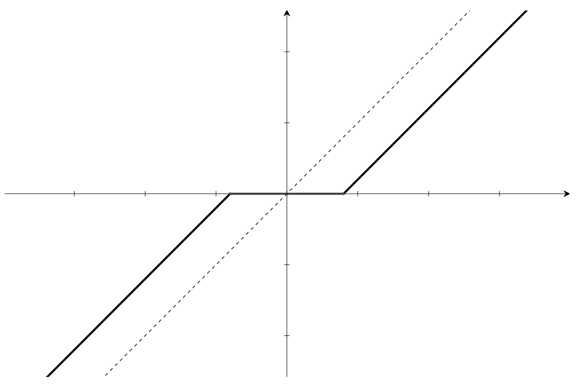
(b) $y = x + \lambda\phi_{\log}'(x)$, for small a,

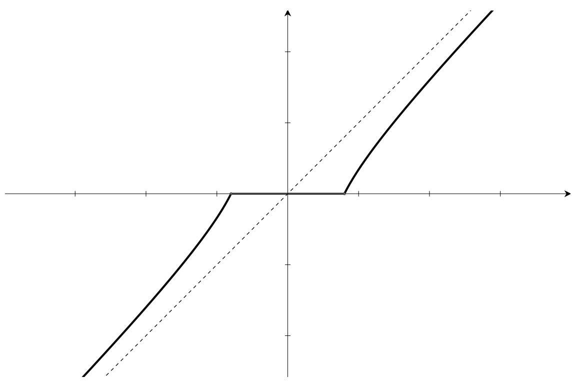
(c) $y = x + \lambda\phi_{\log}'(x)$, for big a,

Figure 2.8: $y = x + \lambda\phi'(x)$.

The threshold function, $\theta(y)$, is computed by exchanging the y and x axes, or by just solving (2.14) with respect to x. $\theta(y)$ for the cases we use the l_1 norm, or the logarithmic penalty function with small a, are illustrated in Figure 2.9a and Figure 2.9b respectively. If a is large, then $\theta_{\log}(y)$ does not define a function of y, as there is not a unique x for each y, as illustrated in Figure 2.9c. When x is not unique, one from its values is the global minimizer of $F(x)$ and the other values are local maxima and minima. So, in that case, $F(x)$ is not strictly convex.

In order $F(x)$ to be strictly convex, $y = x+\lambda\phi_{\log}'(x)$ should be strictly increasing.


(a) $\theta_1(y)$,


(b) $\theta_{\log}(y)$, for small a,

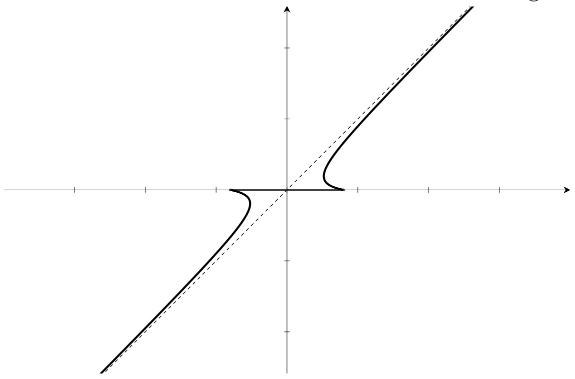

(c) $\theta_{\log}(y)$, for big a,

Figure 2.9: Threshold functions, $\theta(y)$.

That is, the derivative of $x + \lambda\phi'_{\log}(x)$ should be positive for every $x > 0$.

$$\begin{aligned}
\Big(x + \lambda\phi'_{\log}(x)\Big)' &> 0, \qquad \forall x > 0,\\
\phi''_{\log}(x) &> -\frac{1}{\lambda}, \qquad \forall x > 0,\\
-\frac{a}{(1+ax)^2} &> -\frac{1}{\lambda}, \qquad \forall x > 0,\\
a &< \frac{1}{\lambda},
\end{aligned}$$

and since $a > 0$, the range of a, in which $F(x)$ is strictly convex is

$$0 < a < \frac{1}{\lambda}. \tag{2.15}$$

A good penalty function should result in a threshold function with three properties [5] :

- Unbiasedness. The threshold function should not substantially bias (attenuate) large y. When the logarithmic penalty function is used, the value $y - \theta(y)$ decays to zero as y increases. Instead, when the l_1 norm is used, $y - \theta(y)$ is constant and equal to λ. Thus, the logarithmic function is a better penalty function than l_1 norm in the sense of unbiasedness.

- Sparsity. The resulting estimator is a threshold rule which automatically sets small estimated coefficients to zero. Both threshold functions set values $|y| < \lambda$ to zero. So λ in both functions is the threshold value.
- Continuity. When the threshold function is not continuous, it is very sensitive to small changes in its input. Both penalty functions correspond to continuous threshold functions.

2.3.2 Convexity Condition

We will expand the results of the previous section, about the range of a, in which $F(x)$ is strictly convex, in the vector case. That is, the computation of the range of a_i, in which $F(\mathbf{x})$ is strictly convex [1].

We consider the function $v : \mathbb{R} \to \mathbb{R}$ defined as

$$v(x) = \frac{1}{2}x^2 + \lambda\phi(x;a). \tag{2.16}$$

Let $\mathcal{S}$ be the set of pairs (λ, a) for which $v(x)$ is strictly convex. For the logarithmic penalty function, as shown in (2.15), the set $\mathcal{S}$ is given by

$$\mathcal{S} = \left\{(\lambda, a) : \lambda > 0, 0 \le a \le \frac{1}{\lambda}\right\}. \tag{2.17}$$

Let $\mathbf{R}$ be a positive definite diagonal matrix such that $\mathbf{H}^H\mathbf{H} - \mathbf{R}$ is positive semidefinite and r_i denote the i-th diagonal entry of $\mathbf{R}$ (i.e. $[\mathbf{R}]_{i,i} = r_i > 0$). If we add and subtract the term $\frac{1}{2}\mathbf{x}^H\mathbf{R}\mathbf{x}$ in $F(\mathbf{x})$ we have

$$\begin{aligned}
F(\mathbf{x}) &= \frac{1}{2}||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2 + \sum_{i=0}^{n-1}\lambda_i\phi(x_i;a_i) + \frac{1}{2}\mathbf{x}^T\mathbf{R}\mathbf{x} - \frac{1}{2}\mathbf{x}^T\mathbf{R}\mathbf{x} \\
&= \frac{1}{2}\left(\mathbf{y}^H\mathbf{y} - \mathbf{y}^H\mathbf{H}\mathbf{x} - \mathbf{x}^H\mathbf{H}^H\mathbf{y} + \mathbf{x}^H\mathbf{H}^H\mathbf{H}\mathbf{x}\right) + \sum_{i=0}^{n-1}\lambda_i\phi(x_i;a_i) + \frac{1}{2}\mathbf{x}^T\mathbf{R}\mathbf{x} - \frac{1}{2}\mathbf{x}^T\mathbf{R}\mathbf{x} \\
&= \frac{1}{2}\mathbf{x}^H\Big(\mathbf{H}^H\mathbf{H} - \mathbf{R}\Big)\mathbf{x} - \mathbf{y}^H\mathbf{H}\mathbf{x} + \frac{1}{2}\mathbf{y}^H\mathbf{y} + \frac{1}{2}\mathbf{x}^H\mathbf{R}\mathbf{x} + \sum_{i=0}^{n-1}\lambda_i\phi(x_i;a_i).
\end{aligned}$$

We define the function $q(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ as

$$q(\mathbf{x}) = \frac{1}{2}\mathbf{x}^H\Big(\mathbf{H}^H\mathbf{H} - \mathbf{R}\Big)\mathbf{x} - \mathbf{y}^H\mathbf{H}\mathbf{x} + \frac{1}{2}\mathbf{y}^H\mathbf{y}.$$

Since $\mathbf{H}^H\mathbf{H} - \mathbf{R}$ is positive semidefinite, $\frac{1}{2}\mathbf{x}^T\Big(\mathbf{H}^T\mathbf{H} - \mathbf{R}\Big)\mathbf{x}$ is convex. Also, since $\mathbf{y}^T\mathbf{H}\mathbf{x}$ is affine, and $\frac{1}{2}\mathbf{y}^T\mathbf{y}$ is constant, $q(\mathbf{x})$ is convex.

We also define the function $g(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ as

$$
\begin{aligned}
g(\mathbf{x}) &= \frac{1}{2}\mathbf{x}^H\mathbf{R}\mathbf{x} + \sum_{i=0}^{n-1} \lambda_i \phi(x_i; a_i) \\
&= \sum_{i=0}^{n-1} \left(\frac{r_i}{2} x_i^2 + \lambda_i \phi(x_i; a_i) \right) \\
&= \sum_{i=0}^{n-1} r_i \left(\frac{1}{2} x_i^2 + \frac{\lambda_i}{r_i} \phi(x_i; a_i) \right) \\
&= \sum_{i=0}^{n-1} r_i v(x_i),
\end{aligned}
$$

where $v(x)$ was defined in(2.16). If $\left(\frac{\lambda_i}{r_i}, a_i\right) \in \mathcal{S}$, then $g(\mathbf{x})$ is strictly convex.

Thus, $F(\mathbf{x})$ becomes

$$F(\mathbf{x}) = q(\mathbf{x}) + g(\mathbf{x}),$$

and is strictly convex if

$$0 \leq a_i \leq \frac{r_i}{\lambda_i}.$$

A problem that arises here is the computation of the positive definite diagonal matrix $\mathbf{R}$. In order to introduce sparsity more strongly, we seek for big r_i that keep $\mathbf{H}^H\mathbf{H} - \mathbf{R}$ positive semidefinite. This problem is formulated as a semidefinite optimization problem

$$
\begin{aligned}
\max_{\mathbf{r} \in \mathbb{R}^n} \quad & \sum_{i=0}^{n-1} r_i \\
\text{subject to} \quad & r_i \geq c_{min} \\
& \mathbf{H}^H\mathbf{H} - \mathbf{R} \succeq 0, \qquad (2.18)
\end{aligned}
$$

where c_{min} is the minimal eigenvalue of $\mathbf{H}^H\mathbf{H}$ and the inequality $\mathbf{H}^H\mathbf{H} - \mathbf{R} \succeq 0$ means that $\mathbf{H}^H\mathbf{H} - \mathbf{R}$ is positive semidefinite. Since $\mathbf{H}^H\mathbf{H}$ is positive semidefinite, c_{min} is greater or equal to zero. The optimization problem (2.18) is always feasible, since, in the worst case, where every r_i is equal to c_{min}, the eigenvalues of $\mathbf{H}^H\mathbf{H} - c_{min}\mathbf{I}$ are obtained by subtracting c_{min} from the eigenvalues of $\mathbf{H}^H\mathbf{H}$. Thus, $\mathbf{H}^H\mathbf{H} - c_{min}\mathbf{I}$ is positive semidefinite.

2.3.3 Maximally Sparse Convex Algorithm

Based on the previous discussion the forgoing approach of the MSC approach is summarized as follows.

1. Input: $\mathbf{y} \in \mathbb{R}^m$, $\mathbf{H} \in \mathbb{R}^{m \times n}$, $\{\lambda_i > 0, \ \ \forall i = 1, \ldots, n\}$, $\phi : \mathbb{R} \to \mathbb{R}$.

2. Find a positive semidifinite diagonal matrix $\mathbf{R}$ such that $\mathbf{H}^H\mathbf{H} - \mathbf{R}$ is positive semidifinite. (i.e. solve (2.18)).

3. For $i = 1, \ldots, n$, set a_n such that $(\frac{r_i}{\lambda_i}, a_i) \in \mathcal{S}$. For the logarithmic penalty function we have:

$$a_i = \beta \frac{r_i}{\lambda_i}, \quad \text{where } 0 \leq \beta \leq 1.$$

   When $\beta = 0$, the penalty function is simply the l_1 norm. When $\beta = 1$, the penalty function is maximally non-convex (maximally sparsity-inducing). So, β is always set to 1.

4. Minimize (2.12) to obtain $\mathbf{x}$ using a majorization minimization algorithm as shown in the next section.

5. Output: $\mathbf{x} \in \mathbb{R}^n$.

Other penalty functions can be used instead of the logarithmic function, but they must have the property that v in (2.16) is convex for $(\lambda, a) \in \mathcal{S}$ for some set $\mathcal{S}$. For example, $\phi(x, p) = |x|^p$, with $0 < p < 1$, does not qualify.

2.3.4 Iterative Maximally Sparse Convex Algorithm

When $\mathbf{H}$ is nearly singular, $\mathbf{R}$ will be close to zero. This means for the logarithmic penalty function, that r_i is almost zero, which leads a_i to be almost zero. As a consequence, the penalty function is practically the l_1 norm, so the method offers no advantage. In order to broaden the applicability of MSC algorithm, an iterative MSC algorithm is described. In each iteration, MSC algorithm is applied only to the non-zero elements of the sparse solution $\mathbf{x}$, obtained as a result of the previous iteration, and the corresponding columns of $\mathbf{H}$. As the iterations progress, the active columns of $\mathbf{H}$ decreases, $\mathbf{R}$ is less constrained, and eventually r_i becomes greater than zero, which leads a_i to be also greater than zero. Hence, as the iteration progresses the penalty function becomes increasingly non-convex. Therefore, the IMSC algorithm produces a sequence of successively sparser solutions. The procedure can be repeated until there is no change in the index set of non-zero elements. The algorithm can be initialized with the l_1 norm solution, assuming it is reasonably sparse.

The IMSC procedure is described as follows, where $k \geq 1$ denotes the iteration index.

1. Initialization. Find the l_1 norm solution

$$\mathbf{x}^{(1)} = \underset{\mathbf{x} \in \mathbb{R}^n}{\operatorname{argmin}} \quad ||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2 + \sum_{i=0}^{n-1} \lambda_i |x_i|,$$

set $k = 1$ and $K^{(0)} = n$. $\mathbf{H}$ is of size $m \times n$.

2. Identify the non-zero elements of $\mathbf{x}^{(k)}$, and record their indices in the set $\mathcal{K}^{(k)}$

$$\mathcal{K}^{(k)} = \left\{ i \in \mathbb{Z}_n \middle| x_i^{(k)} \neq 0 \right\}.$$

Let $K^{(k)} = |\mathcal{K}^{(k)}|$.

3. Check termination condition: If $K^{(k)}$ is not less than $K^{(k-1)}$, then terminate. The output is $\mathbf{x}^{(k)}$.

4. Define $\mathbf{H}^{(k)}$ as the submatrix of $\mathbf{H}$ containing only columns $l \in \mathcal{K}^{(k)}$. $\mathbf{H}^{(k)}$ is of size $m \times K^{(k)}$. Find a positive semidefinite diagonal matrix $\mathbf{R}^{(k)}$, of size $K^{(k)} \times K^{(k)}$, using (2.18).

5. Set a_i such that $\left(\frac{\lambda_i}{r_i^{(k)}}, a_i\right) \in \mathcal{S},\ \ i \in \mathcal{K}^{(k)}$. For the logarithmic penalty function we have

$$a_i^{(k)} = \frac{r_i^{(k)}}{\lambda_i}, \quad i \in \mathcal{K}^{(k)}.$$

6. Solve the $K^{(k)}$ dimensional convex problem

$$\mathbf{u}^{(k)} = \underset{\mathbf{u} \in \mathbb{R}^{K^{(k)}}}{\operatorname{argmin}} \quad F(\mathbf{u}) = ||\mathbf{y} - \mathbf{H}^{(k)}\mathbf{u}||_2^2 + \sum_{i \in \mathcal{K}^{(k)}} \lambda_i \phi\left(u_i; a_i^{(k)}\right). \tag{2.19}$$

7. Set $\mathbf{x}^{(k+1)}$ as

$$x_i^{(k+1)} = \begin{cases} 0, & \text{for } i \notin \mathcal{K}^{(k)}, \\ u_i^{(k)}, & \text{for } i \in \mathcal{K}^{(k)}. \end{cases}$$

8. Set $k = k + 1$ and go to step 2.

It is obvious that three optimization problems appear in the algorithm. The first one, at step 1, is a least squares problem, regularised with the l_1 norm. The second one, at step 4, is a semidefinite optimization problem, and the last one, at step 6, is a least squares problem, regularised with the a non-convex penalty function.

The first two optimization problems can be easily solved by CVX. CVX is a software package that runs in Matlab, and is used to formulate and solve convex optimization problems by transforming Matlab code into an appropriate modelling language. The description of the optimization problem is typed in a form that looks very similar to how one would write it mathematically on paper, making it easy to use. CVX converts the problem description into an equivalent Linear Program, Quadratic Program, or Semidefinite Program and solves the problem. Model specifications are constructed using common Matlab operations and functions, and standard Matlab code can be freely mixed with these specifications. This combination makes it simple to perform the calculations needed to form optimization problems, or to process the results obtained from their solution.

Unfortunately, CVX is unable to solve the problem (2.19) due to the non-convex nature of the penalty function, $\phi(x)$. Thus, in order to minimize $F(\mathbf{x})$, an iterative algorithm, based on the MM method explained in Section 2.1.9, is used [1].

To apply the MM method to minimize $F(\mathbf{x})$, one may either majorize the term $||\mathbf{y} - \mathbf{H}\mathbf{x}||_2^2$, or the penalty term $\sum_i \phi\left(x\left(i\right)\right)$, or both. We choose to majorize just the penalty term and it will be useful to consider first the scalar case.

We will find a majorizer for $\phi(x)$ with $x \in \mathbb{R}$. The majorizer $g(x)$ should be an upper bound for $\phi(x)$ that coincides with $\phi(x)$ at a specified point $v \in \mathbb{R}$. That is

$$\begin{aligned} g(x) &\geq \phi(x), \quad \forall\, x \in \mathbb{R}, \\ g(v) &= \phi(v). \end{aligned} \tag{2.20}$$

We will use a quadratic function to majorize the penalty function $\phi(x)$, as shown in Figure 2.10. Thus, $g(x)$ will be of the form

$$g(x) = mx^2 + c. \tag{2.21}$$

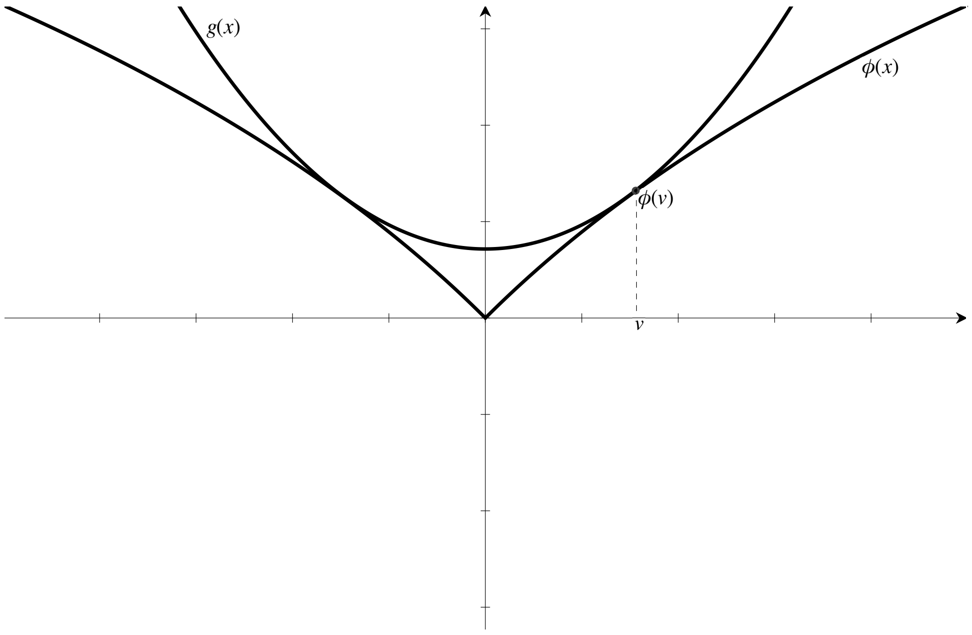


Figure 2.10: $g(x)$ majorizes $\phi(x)$.

Since the quadratic function $g(x)$ is a convex function, for a specified point v, $g(x)$ is greater or equal than the line that passes from v and has slope equal to $\nabla_v g(v)$. Also, since the penalty function $\phi(x)$ is concave for $x > 0$, for a specified point v, $\phi(x)$ is less or equal than the line that passes from v and has slope equal to $\nabla_v \phi(v)$. Since the quadratic function $g(x)$ and the penalty function $\phi(x)$ are symmetric functions, the conditions of a majorizer function in (2.20) can be written for quadratic function case as

$$\begin{aligned} g(v) &= \phi(v), \\ g'(v) &= \phi'(v), \end{aligned} \tag{2.22}$$

Combining (2.21) and (2.22), we compute m and c

$$\begin{aligned} m &= \frac{1}{2v}\phi'(v), \\ c &= \phi(v) - \frac{v}{2}\phi'(v). \end{aligned}$$

The majorizer $g(x)$ is therefore given by

$$g(x) = \frac{1}{2v}\phi'(v)x^2 + \phi(v) - \frac{v}{2}\phi'(v).$$

With this function g, we have $g(x) \geq \phi(x)$ with equality at $x = v$. To emphasize the dependence of $g(x)$ on the point v, we write

$$g(x, v) = \frac{1}{2v}\phi'(v)x^2 + \phi(v) - \frac{v}{2}\phi'(v). \tag{2.23}$$

The scalar majorizer can be used to obtain a majorizer for $F(\mathbf{x})$ in (2.19). For the vector case, a majorizer for $\sum_n \phi\left(x(n)\right)$ is the $\sum_n g(x(n), v(n))$. That is

$$\sum_n g(x(n), v(n)) \geq \sum_n \phi(x(n)), \tag{2.24}$$

with equality if $\mathbf{x} = \mathbf{v}$. The left-hand-side of (2.24) can be written compactly as

$$\sum_n g(x(n), v(n)) = \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + c, \tag{2.25}$$

where

$$\mathbf{W} = \begin{bmatrix} \frac{\phi'(v(1))}{v(1)} & & \\ & \ddots & \\ & & \frac{\phi'(v(N))}{v(N)} \end{bmatrix},$$

and

$$c = \sum_n \phi(v(n)) - \frac{v}{2}\phi'\left(v(n)\right). \tag{2.26}$$

According to (2.24) and (2.25), a majorizer for $F(\mathbf{x})$ is given by

$$G(\mathbf{x}, \mathbf{v}) = \frac{1}{2}||\mathbf{y} - \mathbf{A}\mathbf{x}||_2^2 + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + c. \tag{2.27}$$

Minimizing $G(\mathbf{x}, \mathbf{v})$, with respect to $\mathbf{x}$, we have

$$\begin{aligned}
\nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{v}) &= \nabla_{\mathbf{x}}\Big(\frac{1}{2}(\mathbf{y} - \mathbf{A}\mathbf{x})^T(\mathbf{y} - \mathbf{A}\mathbf{x}) + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + c\Big) \\
&= \nabla_{\mathbf{x}}\Big(\frac{1}{2}(\mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{A}\mathbf{x} - \mathbf{x}^T\mathbf{A}^T\mathbf{y} + \mathbf{x}^T\mathbf{A}^T\mathbf{A}\mathbf{x}) + \frac{1}{2}\mathbf{x}^T\mathbf{W}\mathbf{x} + c\Big) \\
&= -\mathbf{A}^T\mathbf{y} + \mathbf{A}^T\mathbf{A}\mathbf{x} - \mathbf{W}\mathbf{x},
\end{aligned}$$

and by solving $\nabla_{\mathbf{x}} G(\mathbf{x}, \mathbf{v}) = \mathbf{0}$ we obtain

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A} + \mathbf{W})^{-1}\mathbf{A}^T\mathbf{y}, \tag{2.28}$$

where $\mathbf{W}$ depends on $\mathbf{v}$.

Therefore, the MM update with $G_k(\mathbf{x}) = G(\mathbf{x}, \mathbf{x}_k)$ produces the sequence

$$\mathbf{x}_{k+1} = (\mathbf{A}^T\mathbf{A} + \mathbf{W}_k)^{-1}\mathbf{A}^T\mathbf{y}. \tag{2.29}$$

In that case, we have

$$\mathbf{W}_k = \begin{bmatrix} \frac{\phi'(x_k(1))}{x_k(1)} & & \\ & \ddots & \\ & & \frac{\phi'(x_k(N))}{x_k(N)} \end{bmatrix}.$$

The solution $\mathbf{x}_k$ of each iteration is expected to be sparse. Thus, some components of $\mathbf{x}_k$ will go to zero and consequently some entries of $\mathbf{W}_k$ will go to infinity, making (2.29) numerically inaccurate. This problem is avoided by using the Matrix Inversion Lemma [16] to write

$$(\mathbf{A}^T\mathbf{A} + \mathbf{W}_k)^{-1} = \mathbf{W}_k^{-1} - \mathbf{W}_k^{-1}\mathbf{A}^T(\mathbf{I} + \mathbf{A}\mathbf{W}_k^{-1}\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{W}_k^{-1}.$$

As components of $\mathbf{x}_k$ go to zero, the entries of $\mathbf{W}_k^{-1}$ go to zero instead of to infinity. By setting $\mathbf{\Lambda}_k := \mathbf{W}_k^{-1}$ the quadratic MM update becomes

$$\mathbf{x}_{k+1} = \mathbf{\Lambda}_k\mathbf{A}^T\mathbf{y} - \mathbf{\Lambda}_k\mathbf{A}^T(\mathbf{I} + \mathbf{A}\mathbf{\Lambda}_k\mathbf{A}^T)^{-1}\mathbf{A}\mathbf{\Lambda}_k\mathbf{A}^T\mathbf{y}.$$

Chapter 3

Sparse Channel Estimation

The transmission through a sparse multipath channel, as explained earlier, introduces ISI, making the implementation of an equalizer at the receiver necessary. An exact estimate of the sparse multipath channel will greatly improve the performance of the equalization. Thus, the channel estimation problem is of significant importance in communication systems.

Typically, this is accomplished by probing the channel with a known training sequence and processing the channel output.

3.1 Training Sequence

In order to estimate the channel, a sequence of N pseudo-random pilot symbols is sent through the channel, and observed at the receiver. By pseudo-random, we mean that the values the pilot symbols are chosen to be random independent and identically distributed (i.i.d.), but they are known in advance at the receiver.

We assume $\mathbf{x}$ is the training sequence of length N, $\mathbf{h}$ is the channel impulse response of length L, $\mathbf{r}$ is the resulting vector of observations of length $N + L - 1$, which is corrupted by independent Additive White Gaussian Noise vector $\mathbf{w}$. The resulting input-output relation

$$r_n = \sum_{l=0}^{L} h_l x_{n-l} + w_n,$$

can also be expressed as a matrix-vector product. By symmetry, either the training signal or the channel impulse response could be rewritten as a convolution matrix

$$\mathbf{r} = \mathbf{X}\mathbf{h} + \mathbf{w}. \tag{3.1}$$

The goal is to obtain an estimate of the channel impulse response $\hat{\mathbf{h}}$ from knowledge of the observations $\mathbf{r}$ and training signal $\mathbf{x}$.

Assuming that, before and after the transmission of the training sequence, no symbol is transmitted, the matrix $\mathbf{X}$ is of the form

$$\mathbf{X} = \begin{bmatrix} x_1 & & 0 \\ x_2 & \ddots & \\ \vdots & \ddots & x_1 \\ x_N & & x_2 \\ & \ddots & \vdots \\ 0 & & x_N \end{bmatrix}. \tag{3.2}$$

The pause of the transmission before and after the transmission of the training sequence is called guard interval and has length at least $L-1$. When guard interval is not available, the zeros in the convolution matrix in (3.2) would be replaced by the data sequence, thus, the first and last $L-1$ observations contain contributions from the unknown data, rendering them useless for estimation purposes [6].

Therefore, in the absence of guard interval, the training sequence matrix is of the form

$$\mathbf{X} = \begin{bmatrix} x_L & x_{L-1} & \cdots & x_2 & x_1 \\ x_{L+1} & x_L & \cdots & x_3 & x_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{L+M-1} & x_{L+M-2} & \cdots & x_{M+1} & x_M \end{bmatrix},$$

where $M = N - L + 1$.

3.2 Least Squares Estimation

Conventional LS based channel estimation schemes, solves the minimization problem

$$\hat{\mathbf{h}}_{LS} = \underset{\mathbf{h}}{\operatorname{argmin}} \quad ||\mathbf{X}\mathbf{h} - \mathbf{r}||_2,$$

which has a closed form solution

$$\hat{\mathbf{h}}_{LS} = (\mathbf{X}^H\mathbf{X})^{\dagger}\mathbf{X}^H\mathbf{r},$$

where $\mathbf{A}^{\dagger}$ is the pseudoinverse of $\mathbf{A}$.

While appropriate for rich channels, LS estimation ignores the structure of a sparse multipath channel, which leads to poor performance, since estimation effort is directed towards estimating all the channel coefficients, many of which might be zero.

As shown in Figure 3.1, the LS estimator fails to estimate the sparse channel impulse response. The length of channel impulse response is $L = 100$ and the length of the the training sequence is $N = 150$. The SNR is set to 15dB.

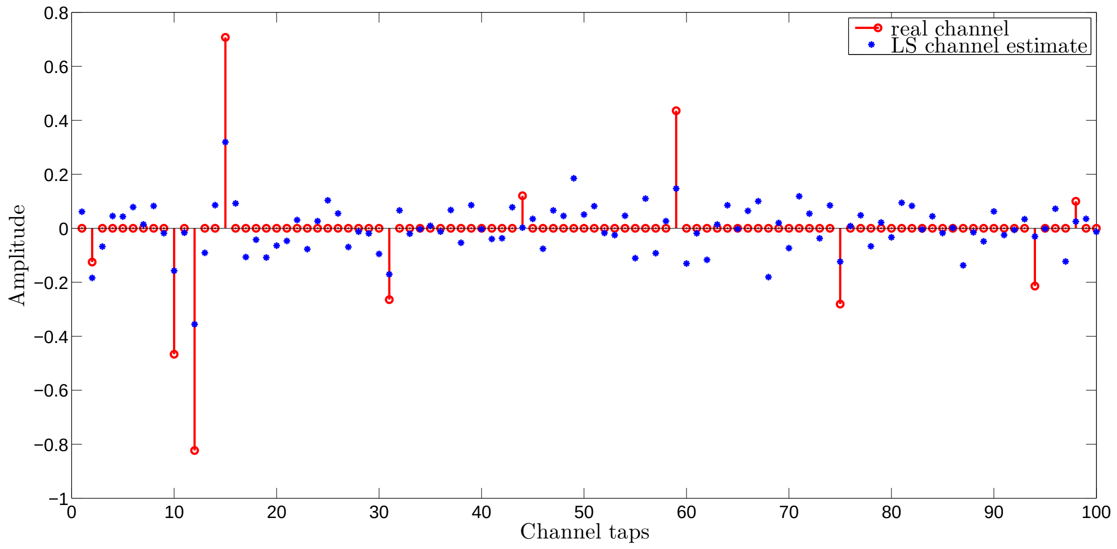


Figure 3.1: Channel estimate with LS estimator.

3.3 Sparse Channel Estimation

Due to the sparse impulse response of the channel, a better estimate will be provided by regularizing with the l_1 norm, as discussed in the previous chapter. The channel estimation problem becomes

$$\hat{\mathbf{h}}_{l_1} = \underset{\mathbf{h}}{\operatorname{argmin}} \quad \frac{1}{2}||\mathbf{Xh} - \mathbf{r}||_2^2 + \lambda||\mathbf{h}||_1. \tag{3.3}$$

A problem that arises here is the computation of a λ that provides good estimates. If we assume that for each element of $\mathbf{h}$ we have a different weight λ_i, we can rewrite (3.3) as

$$\hat{\mathbf{h}}_{l_1} = \underset{\mathbf{h}}{\operatorname{argmin}} \quad F(\mathbf{h}) = \frac{1}{2}||\mathbf{Xh} - \mathbf{r}||_2^2 + \sum \lambda_i \phi_1(h_i), \tag{3.4}$$

where $\phi_1(h_i) = |h_i|$.

When the cost function $F(\mathbf{h})$ in (3.4) is strictly convex and since the penalty function ϕ_1 is differentiable except at zero, then $\mathbf{h}^*$ minimizes $F(\mathbf{h})$ if [11]

$$\begin{cases} [\mathbf{X}^H(\mathbf{r} - \mathbf{Xh}^*)]_i = \lambda_i \phi_1'(h_i), & \text{for } h_i^* \neq 0, \\ \lambda_i \phi_1'(0^-) \leq [\mathbf{X}^H(\mathbf{r} - \mathbf{Xh}^*)]_i \leq \lambda_i \phi_1'(0^+), & \text{for } h_i^* = 0, \end{cases} \tag{3.5}$$

where $[\mathbf{v}]_i$ denotes the i-th component of the vector $\mathbf{v}$.

The condition (3.5) can be used to set the values of λ_i [1]. If $\mathbf{h} = \mathbf{0}$ in (3.1), then $\mathbf{r}$ consists of noise only (i.e. $\mathbf{r} = \mathbf{w}$). Then (3.5) gives

$$\lambda_i \phi_1'(0^-) \leq [\mathbf{X}^H \mathbf{w}]_i \leq \lambda_i \phi_1'(0^+). \tag{3.6}$$

Since $\phi_1(0^-) = -1$ and $\phi_1(0^+) = 1$, (3.6) becomes

$$|[\mathbf{X}^H \mathbf{w}]_i| \leq \lambda_i. \tag{3.7}$$

The larger the λ_i is, the more h_i will be attenuated. Hence, it is reasonable to set λ_i to the smallest value satisfying (3.7):

$$\lambda_i \approx \max |[\mathbf{X}^H \mathbf{w}]_i|. \tag{3.8}$$

Since, $\mathbf{w}$ is unknown in practice, (3.8) can be estimated based on knowledge of statistics of the noise. For example, assuming that $\mathbf{w}$ is white Gaussian noise with variance σ_w^2, we can empirically set λ_i as the product of the standard deviation of $[\mathbf{X}^H \mathbf{w}]_i$ with a small positive number

$$\lambda_i \approx 3\sigma_w ||\mathbf{X}(:, i)||_2, \tag{3.9}$$

where $\mathbf{X}(:, i)$ is the i-th column of $\mathbf{X}$.

If $\mathbf{X}$ is a linear convolution matrix, as in our case, then all the columns of $\mathbf{X}$ have equal norm. So (3.9) becomes

$$\lambda_i = \lambda = 3\sigma_w ||\mathbf{x}||_2.$$

As shown in Figure 3.2, the estimates of the channel impulse response are very accurate. The length of channel impulse response is 100 taps and the length of the the training sequence is 150. The SNR is set to 15dB.

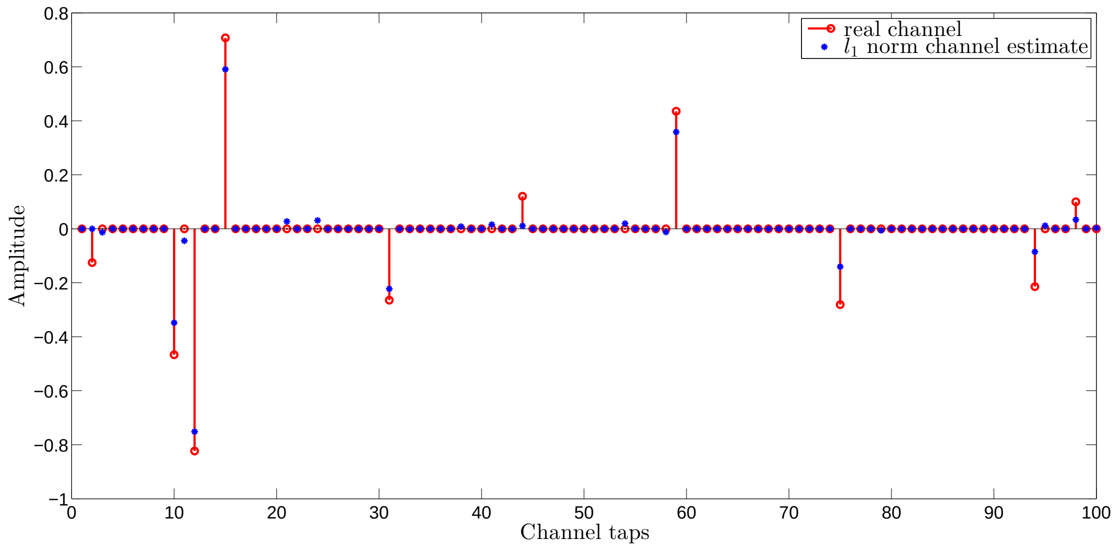


Figure 3.2: Channel estimate with l_1 norm regularized estimator.

In order to achieve even more accurate estimates we could use the MSC algorithm, explained in the previous chapter. The application of the MSC algorithm in the sparse channel estimation problem is straightforward. The estimates of the channel impulse response, acquired by the MSC algorithm, are illustrated in Figure 3.3. The length of channel impulse response is 100 taps and the length of the the training sequence is 150. The SNR is set to 15dB.

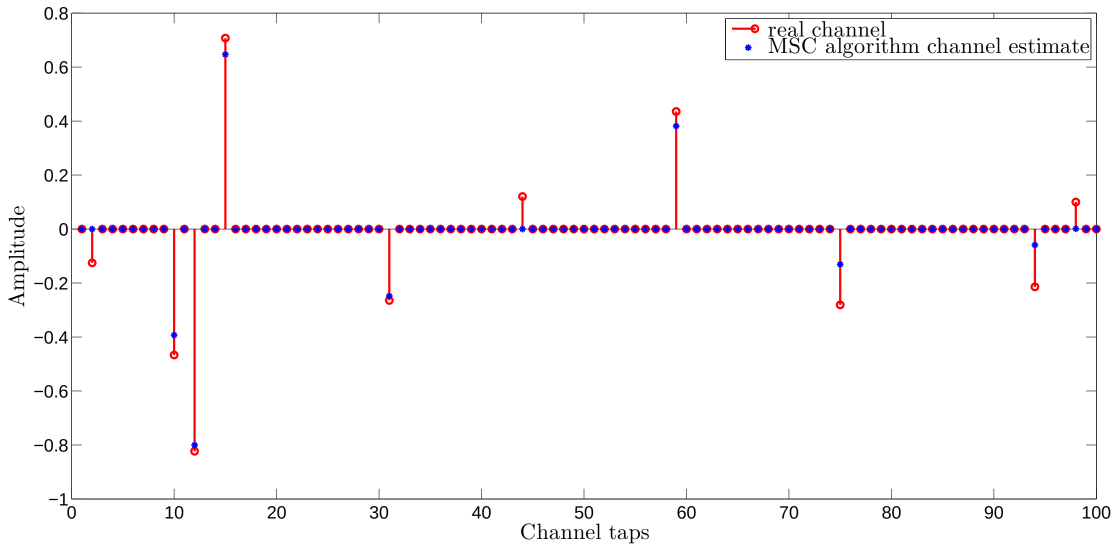


Figure 3.3: Channel estimate with MSC algorithm estimator.

We observe that the estimates of the taps with high amplitude are decreased, because of the bias introduced by the penalty functions. In order to improve the biased estimates, we obtain the support from these estimations and then proceed in solving a LS estimation problem for these positions. One way to obtain the support of the estimations, since usually the knowledge of the exact number of the strong taps of the channel impulse response at the receiver is not possible, is to use a threshold. We set the threshold approximately in the order of the standard deviation of the noise, σ_w, and the taps that exceed that threshold gives us the support of the estimate. Finally we solve a LS estimation problem for the positions of the support. The biased and unbiased estimates of the l_1 norm and MSC estimator are illustrated in Figures 3.4 and 3.5 respectively.

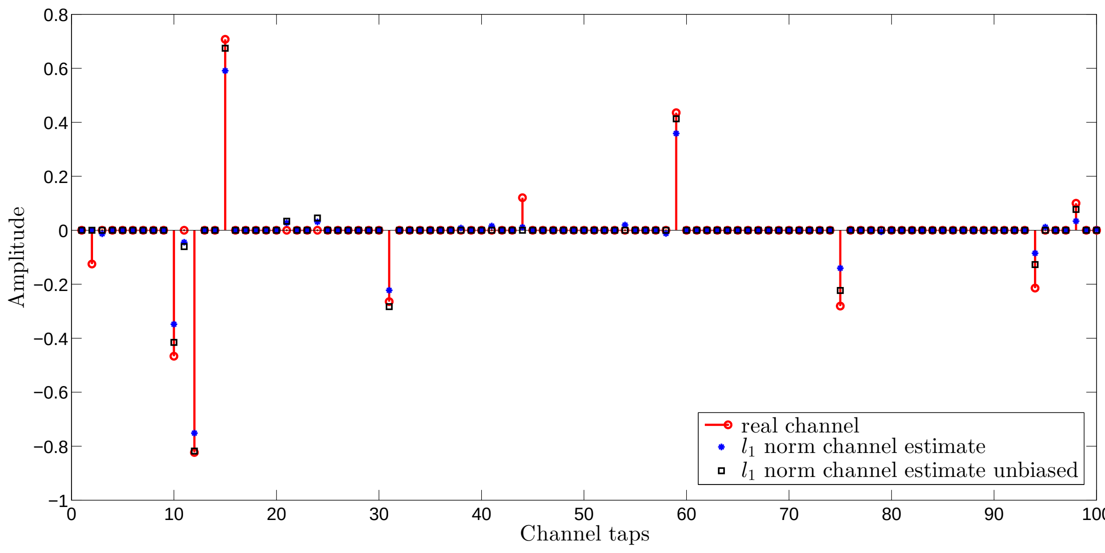


Figure 3.4: Unbiased channel estimate with l_1 estimator.

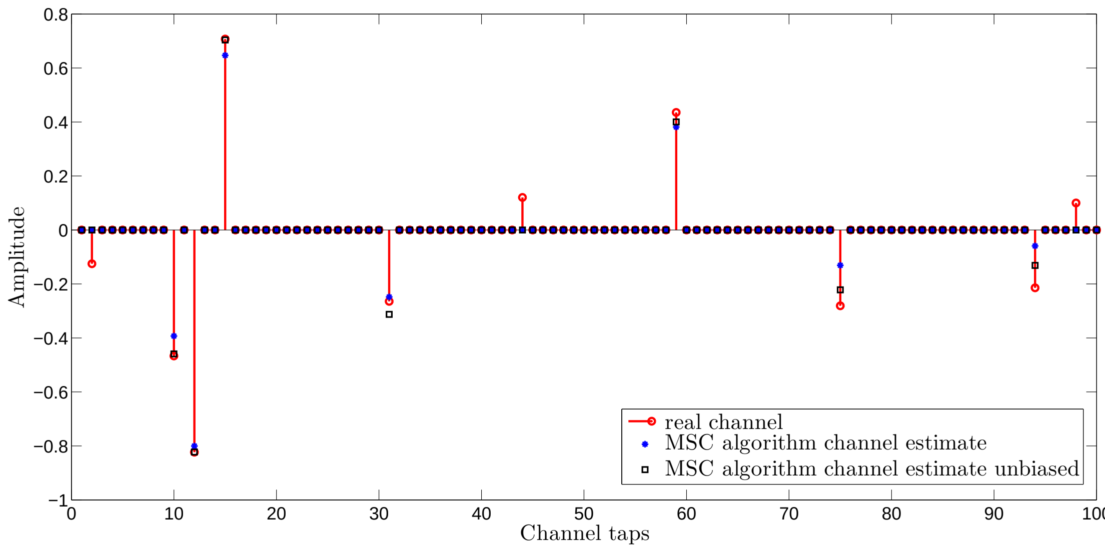


Figure 3.5: Unbiased channel estimate with MSC algorithm estimator.

3.4 Results

To illustrate the performance of the estimators, we compute the MSE by employing the MSC algorithm and the l_1 norm regularized estimator. The estimation error using MSE evaluation criterion can be defined as

$$MSE = E\left\{||\mathbf{h} - \hat{\mathbf{h}}||_2^2\right\}.$$

To evaluate the MSE performance of channel estimators, it is very meaningful to compare their achievements with theoretical performance bounds in practical communication systems [4]. When they reach these bounds, they are approximate optimal and further improvements in these systems are impossible. This motivates the development of lower bounds on the MSE of estimators on sparse channel estimation. Thus, assuming that we know the location set $T = \{i \mid |h_i| > 0\}$ of the dominant channel taps, we define the oracle estimator as

$$\hat{\mathbf{h}}_{oracle} = \left(\mathbf{X}_T^H \mathbf{X}_T\right)^{-1} \mathbf{X}_T^H \mathbf{r}_T, \tag{3.10}$$

where $\mathbf{X}_T$ is the partial training signal constructed from columns of training signal $\mathbf{X}$ corresponding to the dominant taps of the sparse multipath channel vector $\mathbf{h}$.

We compare the performance of the MSC algorithm in the channel estimation and the l_1 norm regularized estimator versus different SNR, which were chosen between 0 and 30. For each SNR we generate 1000 realizations. In each realization we estimate a different channel impulse response of length 100, using a BPSK training sequence of length 150. As a reference, the oracle estimator is also plotted as MSE lower bound. It is seen that, as expected, the performance of all estimators improves with increasing SNR. For low SNR, the two estimators have simillar performance. For high SNR the MSC algorithm exceeds the l_1 norm regularized estimator, and as the SNR increases their performance gap is getting bigger. The results are illustrated in Figure 3.6.

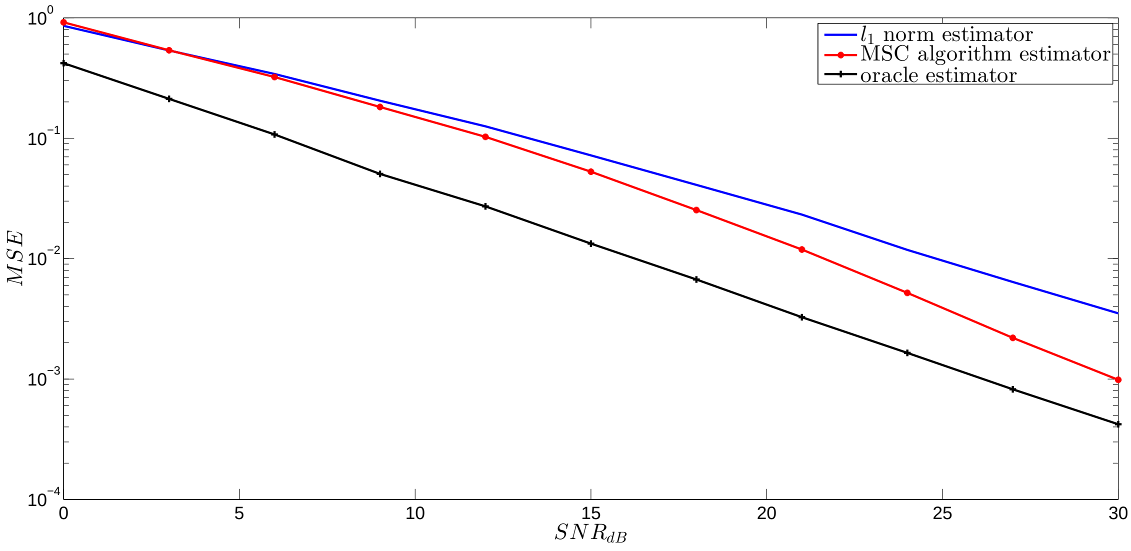


Figure 3.6: MSE performance versus SNR.

We also compare the performance of the estimators versus different length of BPSK training sequence which were chosen between 120 and 180. For each length of training sequence we generate 1000 realization. In each realization we estimate a different channel impulse response of length 100, with the SNR set to 15dB. As a reference, the oracle estimator is also plotted as MSE lower bound. It is seen that, as expected, the performance of all estimators improves with growing number of training sequence. The MSC algorithm provides estimates with better MSE performance for all the available lengths of the training sequence. The results are illustrated in Figure 3.7.

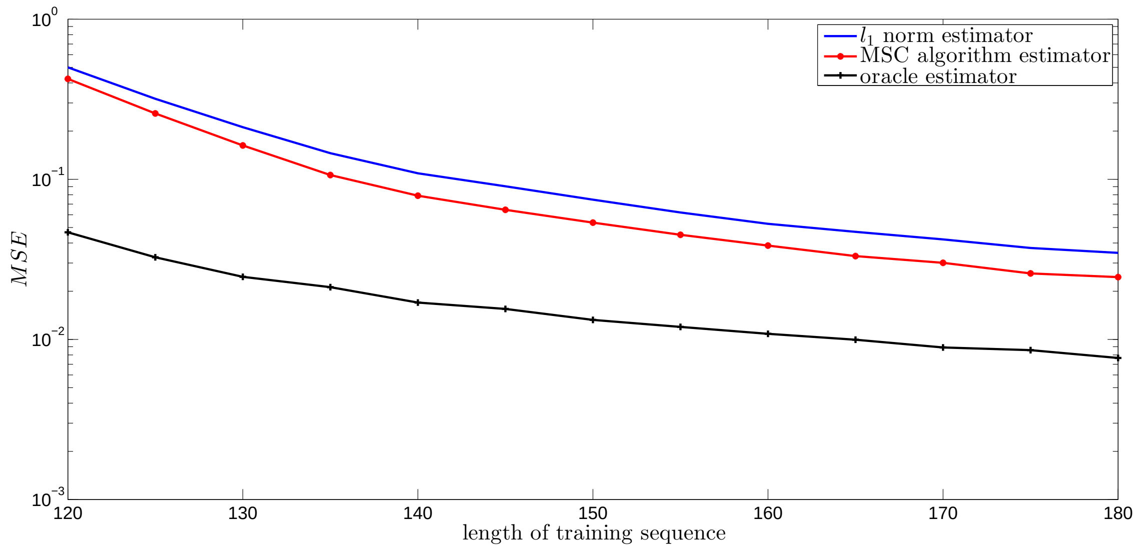


Figure 3.7: MSE performance versus the length of the training sequence.

As explained earlier, we obtain the support from the estimations and then proceed in solving a LS estimation problem for these positions. We repeat the same experiments as above for the unbiased estimations. The accuracy of both estimations is improved, while the MSC algorithm remains superior, showing that it provides a better support than the l_1 norm estimator. The MSE performance versus the SNR and the length of the training sequence are illustrated in Figures 3.8 and 3.9 respectively.

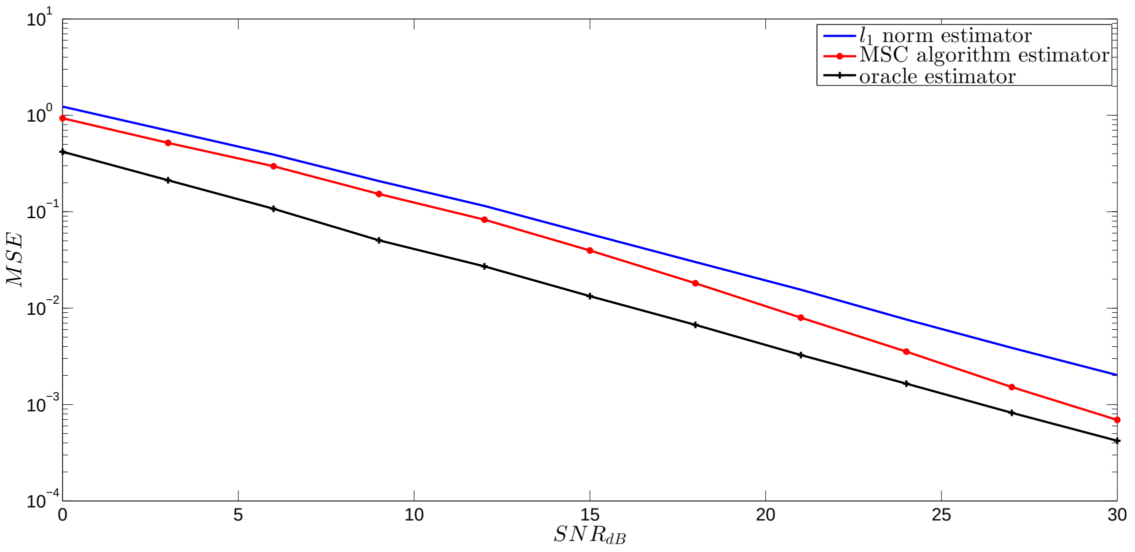


Figure 3.8: MSE performance versus SNR using support.

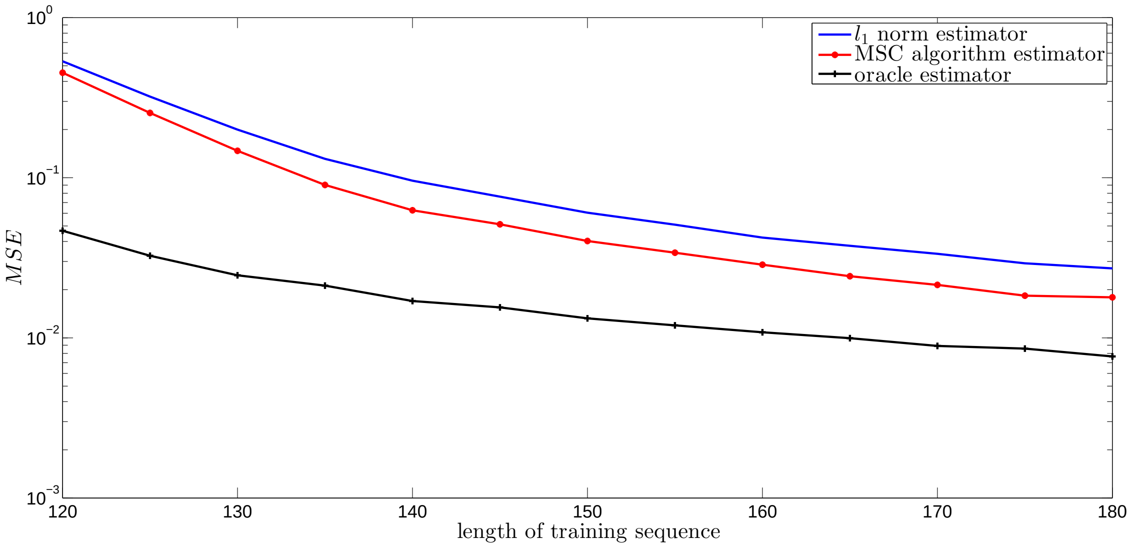


Figure 3.9: MSE performance versus the length of the training sequence using support.

Chapter 4

Decision Feedback Equalizer

In the previous chapter, we estimated the sparse channel impulse response. The accurate estimation is necessary, in order to design an equalizer at the receiver that will cancel the effects of ISI. This kind of equalizer is called channel estimation based, since the coefficients of the filter are computed after the estimation of the channel impulse response. The finite-length MMSE-DFE is a well established ISI mitigating structure on linear, noisy and dispresive channels [3]. A typical block diagram of a DFE is that of Figure 4.1. The DFE consists of the FFF and the FBF. The input of the first filter is the sequence received at the receiver, while the input of the second is the sequence of decisions provided by the detector of the system. Assuming that every equalizer introduces a delay Δ, the FFF is responsible for the cancellation of the ISI produced by the Δ symbols that follow the symbol of interest. On the other hand, the FBF is responsible for the cancellation of the ISI produced by the symbols that reached the receiver before the symbol of interest. In this section, we recapitulate known results concerning the MMSE-DFE. We assume that the channel impulse response is perfectly known.

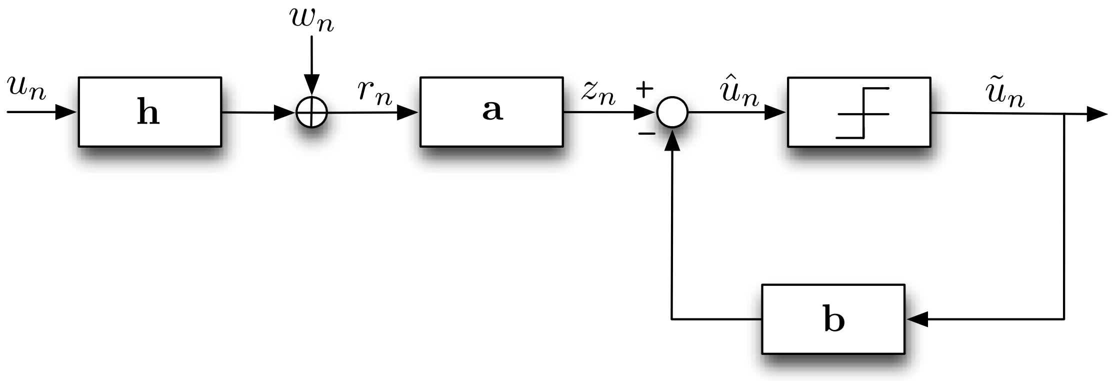


Figure 4.1: Decision feedback equalizer.

4.1 MMSE-DFE

Since the optimization criterion that we will use for the computation of the DFE filters is the minimization of the mean square error, it would be useful to express the error of the equalizer at time n as

$$\begin{aligned} e_n &= u_{n-\Delta} - \hat{u}_n \\ &= u_{n-\Delta} - (\mathbf{a}^H \mathbf{r}_{n:n-N_f+1} - \mathbf{b}^H \tilde{\mathbf{u}}_{n-\Delta-1:n-\Delta-N_b}), \end{aligned}$$

where $\mathbf{r}$ is the received sequence, $\mathbf{a}$ is the FFF of length N_f, $\mathbf{b}$ is the FBF of length N_b, u_n is the i.i.d. input symbols with symbol variance σ_u^2, $\hat{u}_n$ is the pre-detected decision, $\tilde{u}_n$ is the detected decision variables, and Δ is the decision delay introduced by the equalizer.

Assuming all the decisions are correct, $\mathbf{u} = \tilde{\mathbf{u}}$, we get

$$\begin{aligned} e_n &= u_{n-\Delta} - (\mathbf{a}^H \mathbf{r}_{n:n-N_f+1} - \mathbf{b}^H \mathbf{u}_{n-\Delta-1:n-\Delta-N_b}) \\ &= u_{n-\Delta} - \begin{bmatrix} \mathbf{a}^H & -\mathbf{b}^H \end{bmatrix} \begin{bmatrix} \mathbf{r}_{n:n-N_f+1} \\ \mathbf{u}_{n-\Delta-1:n-\Delta-N_b} \end{bmatrix}, \end{aligned}$$

and denoting $\tilde{\mathbf{f}}^H = \begin{bmatrix} \mathbf{a}^H & -\mathbf{b}^H \end{bmatrix}$ the $(N_f + N_b)$-length vector stacking the FFF and FBF tap weights, and $\tilde{\mathbf{r}} = \begin{bmatrix} \mathbf{r}_{n:n-N_f+1} \\ \mathbf{u}_{n-\Delta-1:n-\Delta-N_b} \end{bmatrix}$ we have [2]

$$e_n = u_{n-\Delta} - \tilde{\mathbf{f}}^H \tilde{\mathbf{r}}.$$

The MSE of the equalizer can be expressed as

$$\begin{aligned} MSE &= [|e_n|^2] \\ &= E[|u_{n-\Delta} - \tilde{\mathbf{f}}^H \tilde{\mathbf{r}}|^2] \\ &= E[|u_{n-\Delta}|^2] + \tilde{\mathbf{f}}^H E[\tilde{\mathbf{r}}\tilde{\mathbf{r}}^H]\tilde{\mathbf{f}} - \tilde{\mathbf{f}}^H E[\tilde{\mathbf{r}}\ u_{n-\Delta}^*] - E[u_{n-\Delta}\ \tilde{\mathbf{r}}^H]\tilde{\mathbf{f}}. \end{aligned} \tag{4.1}$$

In order to acquire the correlation matrices needed for the computation of the MSE, we rewrite the input-output relation of the discrete-time baseband equivalent channel, considering a block of N_f output symbols, as

$$\begin{bmatrix} r_n \\ \vdots \\ r_{n-N_f+1} \end{bmatrix} = \begin{bmatrix} h_0 & \cdots & h_L & & & 0 \\ & h_0 & \cdots & h_L & & \\ & & \ddots & & \ddots & \\ 0 & & & h_0 & \cdots & h_L \end{bmatrix} \begin{bmatrix} u_n \\ \vdots \\ u_{n-N_f-L+1} \end{bmatrix} + \begin{bmatrix} w_n \\ \vdots \\ w_{n-N_f+1} \end{bmatrix}, \tag{4.2}$$

where $\mathbf{h} = [h_0 \dots h_L]^T$, is the channel impulse response of length $L+1$, and $\mathbf{w}$ is the (independent of the data sequence) additive white Gaussian noise with variance σ_w^2. The input-output relation (4.2) can be expressed more compactly as

$$\mathbf{r}_{n:n-N_f+1} = \mathbf{H}\mathbf{u}_{n:n-N_f-L+1} + \mathbf{w}_{n:n-N_f+1}. \tag{4.3}$$

The first term of (4.1) is computed as

$$E[|u_{n-\Delta}|^2] = E[|u_n|^2] = \sigma_u^2.$$

The $(N_f + N_b) \times (N_f + N_b)$ auto-correlation matrix appearing in the second term of (4.1) is given by

$$\mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}} = \begin{bmatrix} \mathbf{R}_{\mathbf{rr}} & \sigma_u^2 \mathbf{H}\mathbf{J}_\Delta \\ \sigma_u^2 \mathbf{J}_\Delta^H \mathbf{H}^H & \sigma_u^2 \mathbf{I}_{N_b} \end{bmatrix},$$

where

$$\mathbf{R}_{\mathbf{rr}} = \sigma_w^2 \mathbf{H}\mathbf{H}^H + \mathbf{R}_{\mathbf{ww}},$$

and

$$\mathbf{J}_\Delta = \begin{cases} \begin{bmatrix} \mathbf{0}_{(\Delta+1)\times N_b} \\ \mathbf{I}_{N_b} \\ \mathbf{0}_{s-\Delta\times N_b} \end{bmatrix}, & \text{for } \Delta \le s, \\ \begin{bmatrix} \mathbf{0}_{(\Delta+1)\times N_b} & \\ \mathbf{I}_{s+N_b-\Delta} & \mathbf{0}_{(s-N_b-\Delta)\times(\Delta-s)} \end{bmatrix}, & \text{for } \Delta > s, \end{cases}$$

with $s = N_f + L - N_b - 1$.

The cross-correlation appearing in the third term of (4.1) is given by

$$E[\tilde{\mathbf{r}}\ u_{n-\Delta}^*] = \begin{bmatrix} \sigma_u^2 \mathbf{H} \\ \sigma_u^2 \mathbf{J}_\Delta \end{bmatrix} \mathbf{e}_\Delta.$$

We denote $\tilde{\mathbf{p}}_\Delta = \begin{bmatrix} \sigma_u^2 \mathbf{H} \\ \sigma_u^2 \mathbf{J}_\Delta \end{bmatrix} \mathbf{e}_\Delta$.

All the correlation matrices presented in this section are computed analytically in Appendix A.

Now that we obtained all the correlation matrices needed, we can proceed to the computation of the MSE as

$$MSE = \sigma_u^2 + \tilde{\mathbf{f}}^H \mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}} \tilde{\mathbf{f}} - \tilde{\mathbf{f}}^H \tilde{\mathbf{p}}_\Delta - \tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{f}}. \tag{4.4}$$

We observe that the MSE in (4.4) is a convex function of $\tilde{\mathbf{f}}$. Assuming that $\mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}}$ is non-singular, the optimal $\tilde{\mathbf{f}}$ is obtained by

$$\begin{aligned} \nabla_{\tilde{\mathbf{f}}} MSE &= \mathbf{0} \\ 2\mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}} \tilde{\mathbf{f}}_{opt} - 2\tilde{\mathbf{p}}_\Delta &= \mathbf{0} \\ \tilde{\mathbf{f}}_{opt} &= \mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}}^{-1} \tilde{\mathbf{p}}_\Delta. \end{aligned} \tag{4.5}$$

The optimal FFF and FBF are obteained from $\tilde{\mathbf{f}}$ as follows

$$\begin{aligned} \mathbf{a}_{opt} &= \tilde{\mathbf{f}}_{1:N_f}, \\ \mathbf{b}_{opt} &= \tilde{\mathbf{f}}_{N_f+1:N_f+N_b}. \end{aligned}$$

Substituting the $\tilde{\mathbf{f}}$ of (4.5) in (4.1) we obtain the MMSE:

$$\begin{aligned} MMSE &= \sigma_u^2 + \tilde{\mathbf{f}}_{opt}^H \mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}} \tilde{\mathbf{f}}_{opt} - \tilde{\mathbf{f}}_{opt}^H \tilde{\mathbf{p}}_\Delta - \tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{f}}_{opt} \\ &= \sigma_u^2 - \tilde{\mathbf{p}}_\Delta^H \mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}}^{-1} \tilde{\mathbf{p}}_\Delta. \end{aligned}$$

4.2 Sparse DFE

As explained in chapter 2, a sparse equalizer has many benefits in the equalization of sparse multipath channels. The computation of the coefficients of a sparse equalizer can be approached as a sparsity-performance trade off. As an equalizer is getting more and more sparse, its performance is decreasing, and vice versa. In this section, a convex optimization based approach to design a sparse DFE is presented [2]. A convex optimization based solution for sparse DFE is formulated given a maximum allowable loss in the MSE.

Since $\mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}}$ from the previous section is symmetric, and we have already assumed that is non-singular, we can define the Cholesky factorization [16] of $\mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}}$ as $\mathbf{R}_{\tilde{r}\tilde{r}} = \tilde{\mathbf{L}}\tilde{\mathbf{L}}^H$, where $\tilde{\mathbf{L}}$ is a an $(N_f \times N_f)$ lower triangular matrix, and rewrite equation (4.1) as

$$MSE = \sigma_u^2 + \tilde{\mathbf{f}}^H \tilde{\mathbf{L}}\tilde{\mathbf{L}}^H \tilde{\mathbf{f}} - \tilde{\mathbf{f}}^H \tilde{\mathbf{L}}\tilde{\mathbf{L}}^{-1} \tilde{\mathbf{p}}_\Delta - \tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{L}}^{-H} \tilde{\mathbf{L}}^H \tilde{\mathbf{f}}, \tag{4.6}$$

where $(.)^{-H} = ((.)^H)^{-1}$.

If we add and subtract $\tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{L}}^{-H} \tilde{\mathbf{L}}^H \tilde{\mathbf{p}}_\Delta$ in (4.6), we have

$$\begin{aligned} MSE &= \sigma_u^2 + \tilde{\mathbf{f}}^H \tilde{\mathbf{L}} \tilde{\mathbf{L}}^H \tilde{\mathbf{f}} - \tilde{\mathbf{f}}^H \tilde{\mathbf{L}} \tilde{\mathbf{L}}^{-1} \tilde{\mathbf{p}}_\Delta - \tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{L}}^{-H} \tilde{\mathbf{L}}^H \tilde{\mathbf{f}} + \tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{L}}^{-H} \tilde{\mathbf{L}}^H \tilde{\mathbf{p}}_\Delta - \tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{L}}^{-H} \tilde{\mathbf{L}}^H \tilde{\mathbf{p}}_\Delta \\ &= \left(\sigma_u^2 - \tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{L}}^{-H} \tilde{\mathbf{L}}^H \tilde{\mathbf{p}}_\Delta\right) + \left(\tilde{\mathbf{f}}^H \tilde{\mathbf{L}} \tilde{\mathbf{L}}^H \tilde{\mathbf{f}} - \tilde{\mathbf{f}}^H \tilde{\mathbf{L}} \tilde{\mathbf{L}}^{-1} \tilde{\mathbf{p}}_\Delta - \tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{L}}^{-H} \tilde{\mathbf{L}}^H \tilde{\mathbf{f}} + \tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{L}}^{-H} \tilde{\mathbf{L}}^H \tilde{\mathbf{p}}_\Delta\right) \\ &= \sigma_u^2 - \tilde{\mathbf{p}}_\Delta^H \tilde{\mathbf{L}}^{-H} \tilde{\mathbf{L}}^H \tilde{\mathbf{p}}_\Delta + ||\tilde{\mathbf{L}}^H \tilde{\mathbf{f}} - \tilde{\mathbf{L}}^{-1} \tilde{\mathbf{p}}_\Delta||_2^2 \\ &= MMSE + ||\tilde{\mathbf{L}}^H \tilde{\mathbf{f}} - \tilde{\mathbf{L}}^{-1} \tilde{\mathbf{p}}_\Delta||_2^2. \end{aligned} \tag{4.7}$$

We define the term $||\tilde{\mathbf{L}}^H \tilde{\mathbf{f}} - \tilde{\mathbf{L}}^{-1} \tilde{\mathbf{p}}_\Delta||_2^2$ as MSE_{excess}.

It is obvious from (4.7) that $\tilde{\mathbf{f}}$ controls the MSE only via the MSE_{excess}, since $MMSE$ does not depend on $\tilde{\mathbf{f}}$. If the minimum value of MSE_{excess} is achieved, since $MSE_{excess} \geq 0$, then $MSE = MMSE$. The combined FFF and FBF that achieves that is the $\tilde{\mathbf{f}}_{opt}$. For any other selection of $\tilde{\mathbf{f}}$ we have $MSE_{excess} > 0$ and consequently $MSE > MMSE$, which translates into performance degradation.

In general, $\tilde{\mathbf{f}}_{opt}$ is not sparse, hence, its implementation complexity is high. A practical performance-complexity trade-off can be achieved if we design a sparse $\tilde{\mathbf{f}}$ such that $MSE_{excess} \leq \epsilon$, where ϵ is a small positive number which controls the tolerable performance loss in terms of MSE increase. This can be achieved by solving the optimization problem

$$\begin{aligned} \min_{\tilde{\mathbf{f}}} \quad & ||\tilde{\mathbf{f}}||_1 \\ \text{subject to} \quad & ||\tilde{\mathbf{L}}^H \tilde{\mathbf{f}} - \tilde{\mathbf{L}}^{-1} \tilde{\mathbf{p}}_\Delta||_2^2 \leq \epsilon. \end{aligned} \tag{4.8}$$

The objective function in this minimization is convex, and the constraints define a convex set. Thus, this is a convex optimization problem. From this, we know that any local minimizer of the objective subject to the constraints will also be global minimizer.

4.3 MSC application on Sparse DFE

The promising results of the MSC algorithm in the sparse approximation problem, presented in Section 2, has motivated us to search the potentials of the use of this algorithm for the computation of a sparse DFE. Our goal is to observe how much sparsity will be gained by the use of this algorithm, and how this algorithm will affect the performance of the equalization.

The MSC algorithm, as explained earlier, solves a least squares regularized optimization problem. The form of the optimization problem (4.8) is quite different. In order to use the MSC algorithm for the computation of a sparse DFE, we need to reformulate (4.8) as an unconstrained least squares problem, regularized by the

l_1 norm

$$\min_{\tilde{\mathbf{f}}} \quad \frac{1}{2}||\tilde{\mathbf{L}}^H\tilde{\mathbf{f}} - \tilde{\mathbf{L}}^{-1}\tilde{\mathbf{p}}_\Delta||_2^2 + \lambda||\tilde{\mathbf{f}}||_1, \tag{4.9}$$

for some $\lambda > 0$.

To do so, we have to prove that the optimization problems (4.8) and (4.9) are equivalent, and that there is relation between ϵ and λ. This is proven by showing that there exists a $\tilde{\mathbf{f}}_*$ that satisfies the KKT conditions of problems (4.8) and (4.9). If we define $\mathbf{b} = \tilde{\mathbf{L}}^{-1}\tilde{\mathbf{p}}_\Delta$ and $\mathbf{A} = \tilde{\mathbf{L}}^H$, and take the KKT condition for the optimization problem (4.9) we have

$$\nabla_{\tilde{\mathbf{f}}}\left(\frac{1}{2}||\mathbf{A}\tilde{\mathbf{f}}_* - \mathbf{b}||_2^2 + \lambda||\tilde{\mathbf{f}}_*||_1\right) = \mathbf{0}. \tag{4.10}$$

Taking the KKT condition for the optimization problem (4.8) we have

$$\begin{aligned} \partial_{\tilde{\mathbf{f}}}||\tilde{\mathbf{f}}_*||_1 + \sum_i \lambda_* \nabla_{\tilde{\mathbf{f}}}\left(||\mathbf{A}\tilde{\mathbf{f}}_* - \mathbf{b}||_2^2 - \epsilon\right) &\ni \mathbf{0}, & (4.11)\\ ||\mathbf{A}\tilde{\mathbf{f}}_* - \mathbf{b}||_2^2 - \epsilon &\leq 0, & (4.12)\\ \lambda_* &\geq 0, \\ \lambda_*\left(||\mathbf{A}\tilde{\mathbf{f}}_* - \mathbf{b}||_2^2 - \epsilon\right) &= 0. \end{aligned}$$

The optimality conditions of (4.10) can be rewritten as

$$\begin{aligned} \frac{1}{2}\nabla_{\tilde{\mathbf{f}}}||\mathbf{A}\tilde{\mathbf{f}}_* - \mathbf{b}||_2^2 + \lambda\partial_{\tilde{\mathbf{f}}}||\tilde{\mathbf{f}}_*||_1 &\ni \mathbf{0} \\ \left(\mathbf{A}^H\mathbf{A}\tilde{\mathbf{f}}_* - \mathbf{A}^H\mathbf{b}\right) + \lambda\partial_{\tilde{\mathbf{f}}}||\tilde{\mathbf{f}}_*||_1 &\ni \mathbf{0}. & (4.13) \end{aligned}$$

Also, the optimality conditions of (4.11) can be rewritten as

$$2\lambda_*\left(\mathbf{A}^H\mathbf{A}\tilde{\mathbf{f}}_* - \mathbf{A}^H\mathbf{b}\right) + \partial_{\tilde{\mathbf{f}}}||\tilde{\mathbf{f}}_*||_1 \ni \mathbf{0}. \tag{4.14}$$

Assuming that the inequality constraint in (4.12) is active, then $\lambda_* > 0$, so (4.14) becomes

$$\left(\mathbf{A}^H\mathbf{A}\tilde{\mathbf{f}}_* - \mathbf{A}^H\mathbf{b}\right) + \frac{1}{2\lambda_*}\partial_{\tilde{\mathbf{f}}}||\tilde{\mathbf{f}}_*||_1 \ni \mathbf{0} \tag{4.15}$$

It is obvious that if

$$\lambda = \frac{1}{2\lambda_*}, \tag{4.16}$$

then (4.13) and (4.15) are the same. Thus, there is an $\tilde{\mathbf{f}}_*$ that solves both problems (4.8) and (4.9).

Since we have proven that we can rewrite the optimization problem (4.8) as a l_1 norm regularized least squares problem, as in (4.9), we can replace the l_1 norm with the logarithmic penalty function. Our optimization problem then becomes

$$\min_{\tilde{\mathbf{f}}} \quad \frac{1}{2}||\tilde{\mathbf{L}}^H\tilde{\mathbf{f}} - \tilde{\mathbf{L}}^{-1}\tilde{\mathbf{p}}_\Delta||_2^2 + \lambda \sum_{i=0}^{n-1} \phi(\tilde{f}_i; a_i), \tag{4.17}$$

and we can apply the MSC algorithm.

4.4 Results

In this section, we consider the optimization problems (4.8) and (4.17) in order to obtain DFE filters. Our goal is to examine if the MSC algorithm could provide more sparse filters with better performance, in terms of Bit Error Rate (BER) and MSE.

Thus, we compare the BER, the MSE and the number of non-zero taps of the DFE filters acquired by the MSC algorithm and the l_1 norm, versus different SNR, which were chosen between 0 and 30. For each SNR we generate 10000 realizations, and in each one we send 700 BPSK symbols. The length of the channel impulse response used is 100.

We have shown that the MSE can be expressed as a sum of the MMSE and the MSE_{excess}. The MMSE depends on the SNR, and the MSE_{excess} is controlled by a small positive number, ϵ. If we use the same ϵ to control the MSE_{excess} for all the available SNRs, it might not always result in a sparse solution. In order to control better the level of sparsity of the solutions for all the SNRs, we use a percentage of the MMSE as ϵ. That is

$$\epsilon = \epsilon' \; MMSE,$$

where $0 < \epsilon' < 1$.

A small ϵ' results in a solution that is very close to the MMSE-DFE, and will not be very sparse. As we increase ϵ' the sparsity increases as well. In order to examine in which cases the MSC algorithm has an advantage, we will use three values for ϵ' throughout our experiments.

Once we choose the ϵ', we proceed in solving the optimization problem (4.8).

The performance and the sparsity of the resulting DFE filters are studied and used as a reference for the MSC-DFE filters. The solution is obtained using CVX, which also returns the dual optimal λ_*. As shown in (4.16), λ_* helps us determine the λ that we will use in the optimization problem (4.17).

For $\epsilon' = 0.01$, as shown in Figure 4.4, the MSC algorithm provides more sparse DFE filters. Also, as shown in Figure 4.3 and Figure 4.2, the performance of the two resulting DFEs are very similar, giving an advantage to the MSC algorithm.

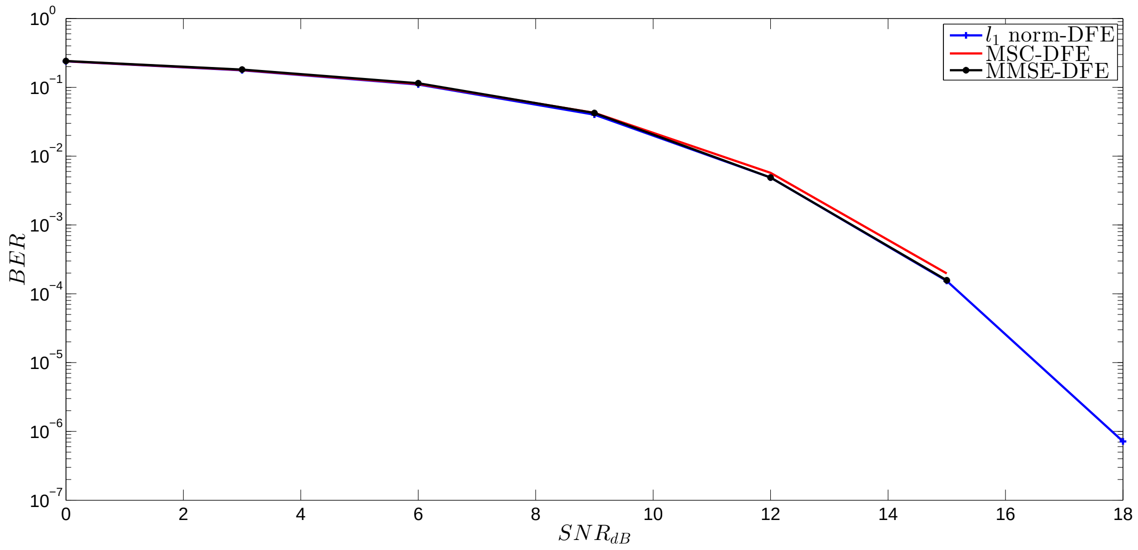


Figure 4.2: Bit error rate for $\epsilon' = 0.01$.

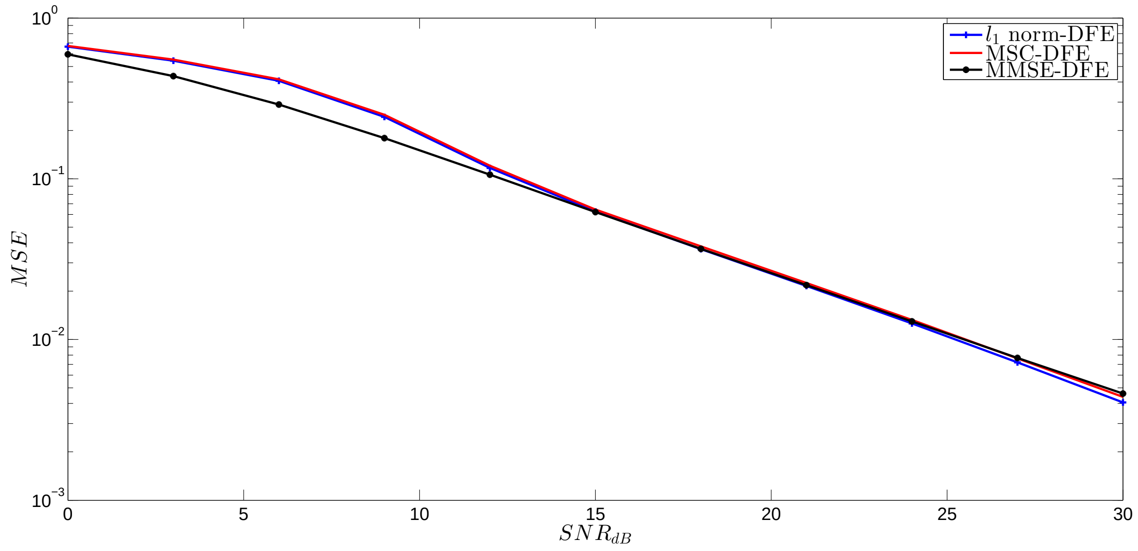


Figure 4.3: Mean square error for $\epsilon' = 0.01$.

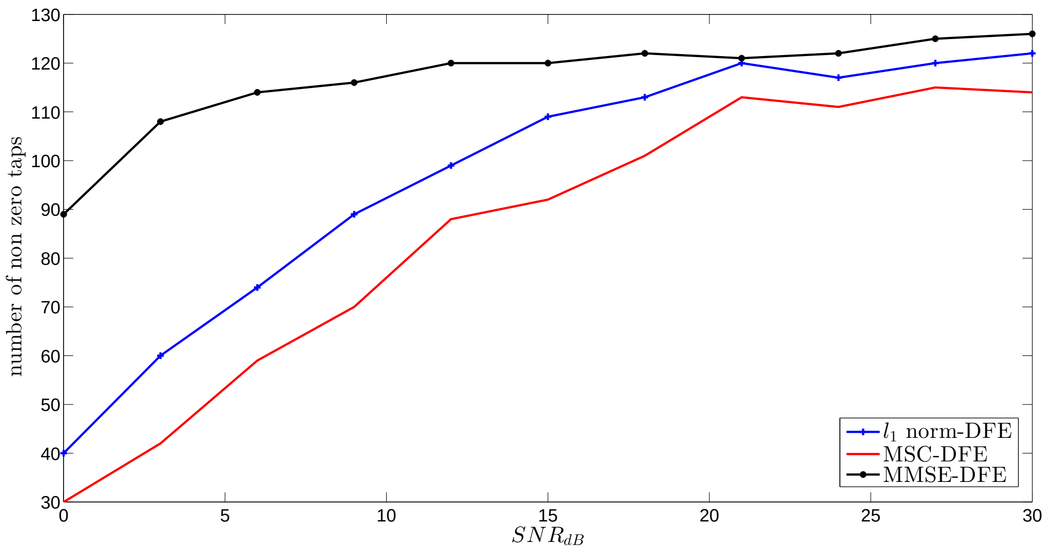


Figure 4.4: Tap support for $\epsilon' = 0.01$.

For $\epsilon' = 0.1$ the MSC algorithm obtains much more sparse filters, as shown in Figure 4.7, but the performance is slightly degraded, as shown in Figure 4.5 and Figure 4.6.

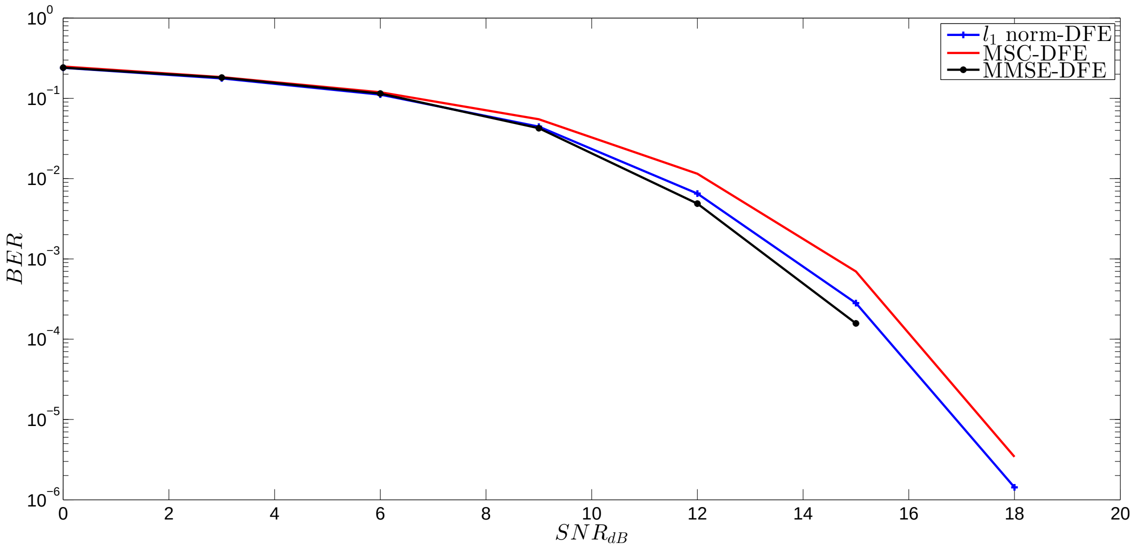


Figure 4.5: Bit error rate for $\epsilon' = 0.1$.

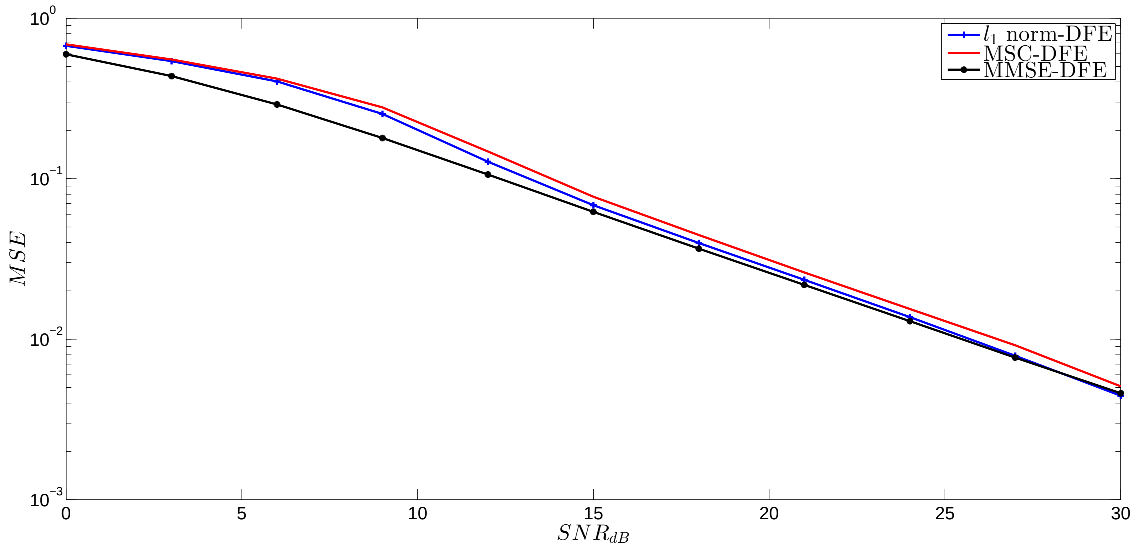


Figure 4.6: Mean square error for $\epsilon' = 0.1$.

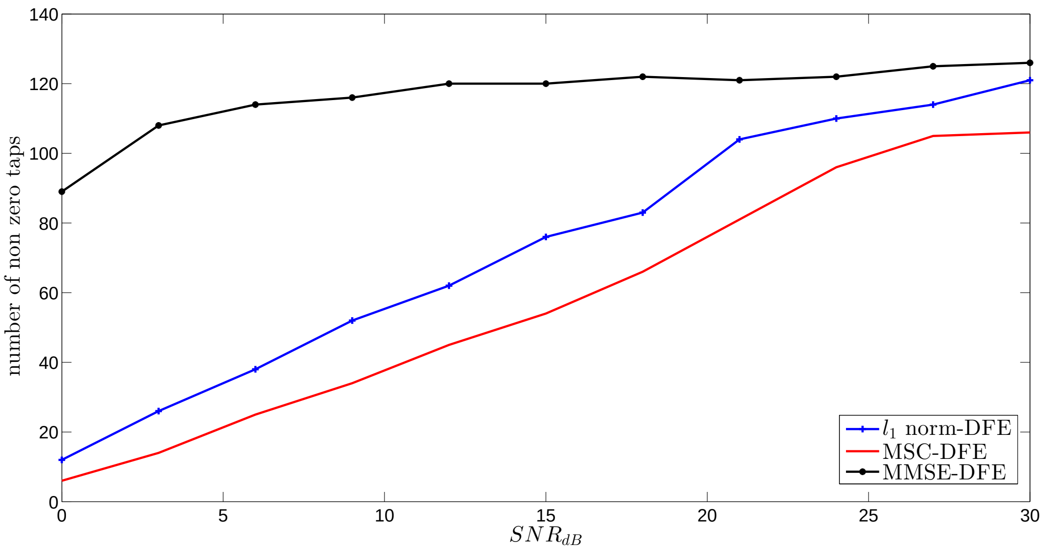


Figure 4.7: Tap support for $\epsilon' = 0.1$.

In order to make a more fair comparison, we adjust the λ, used in the MSC algorithm, in order to provide us a less sparse solution, but with similar performance with the l_1 norm. In that case, as shown in Figures 4.8, 4.9 and 4.10, we achieved similar performance and a more sparse filter.

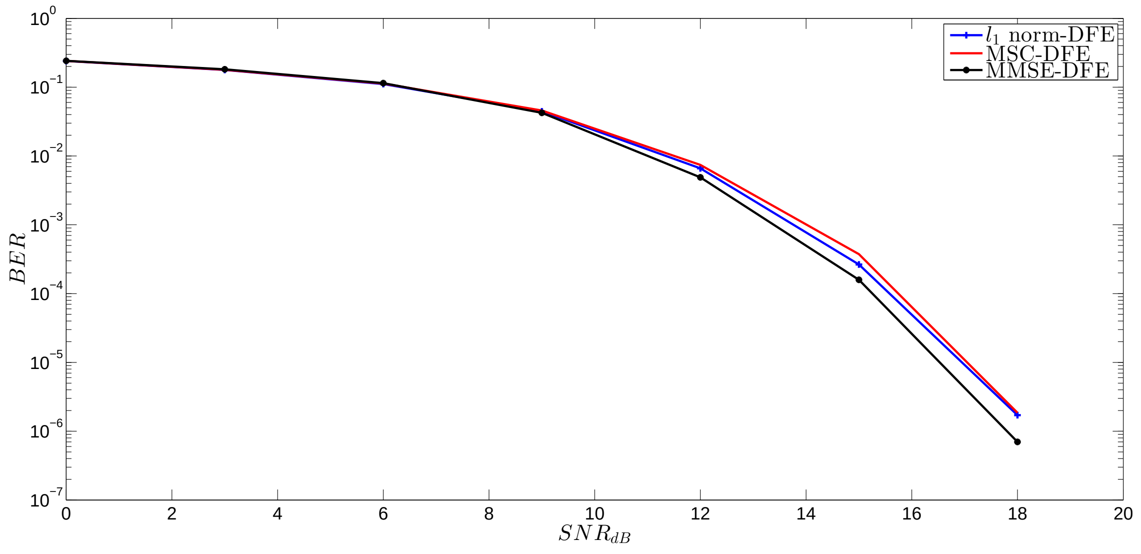


Figure 4.8: Bit error rate for $\epsilon' = 0.1$ and adjusting λ.

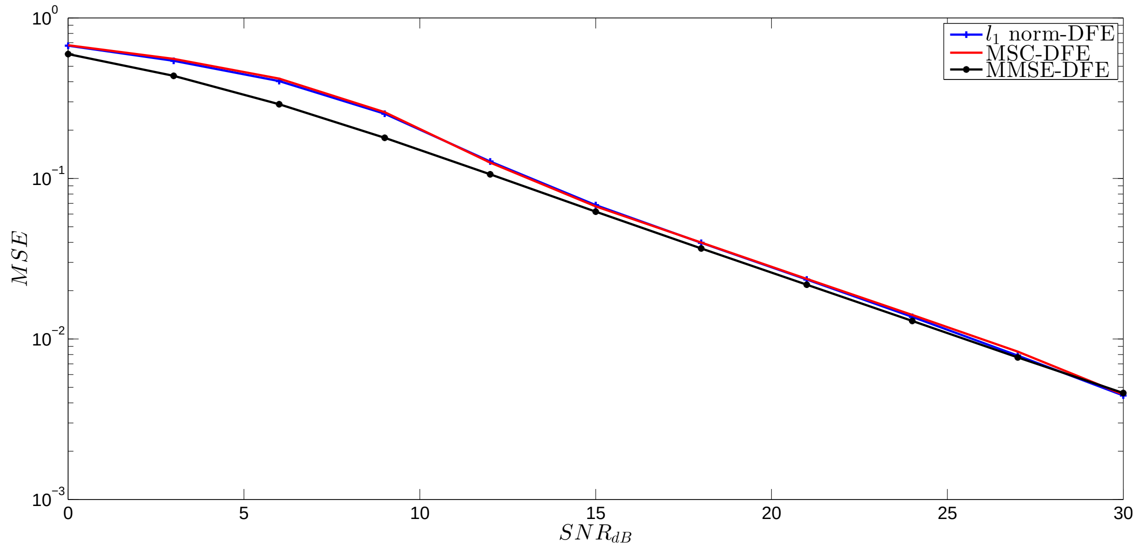


Figure 4.9: Mean square error for $\epsilon' = 0.1$ and adjusting λ.

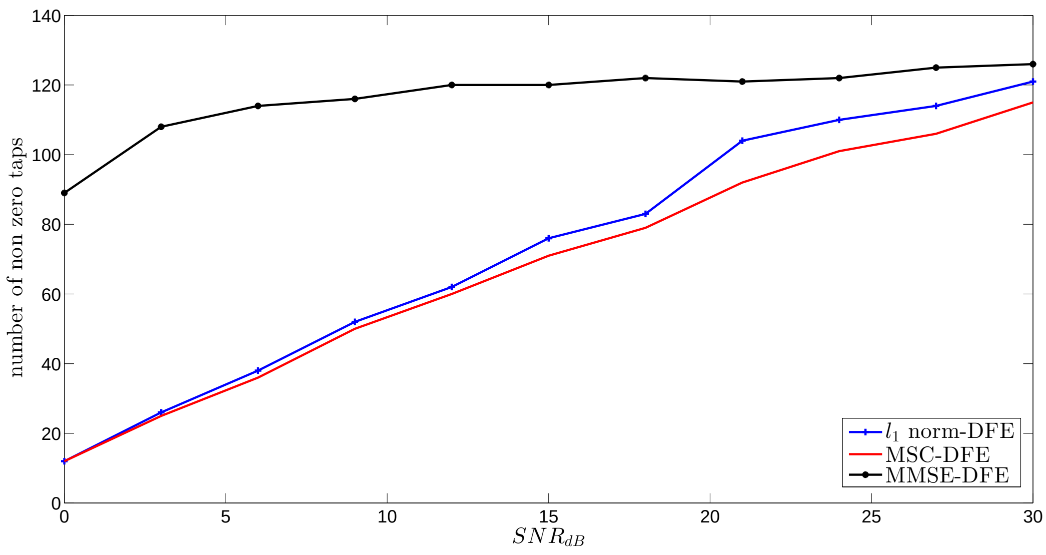


Figure 4.10: Tap support for $\epsilon' = 0.1$ and adjusting λ.

For $\epsilon' = 0.4$, as shown in Figure 4.11, Figure 4.12 and Figure 4.13, the MSC algorithm obtains more sparse filters with similar performance with the l_1 norm.

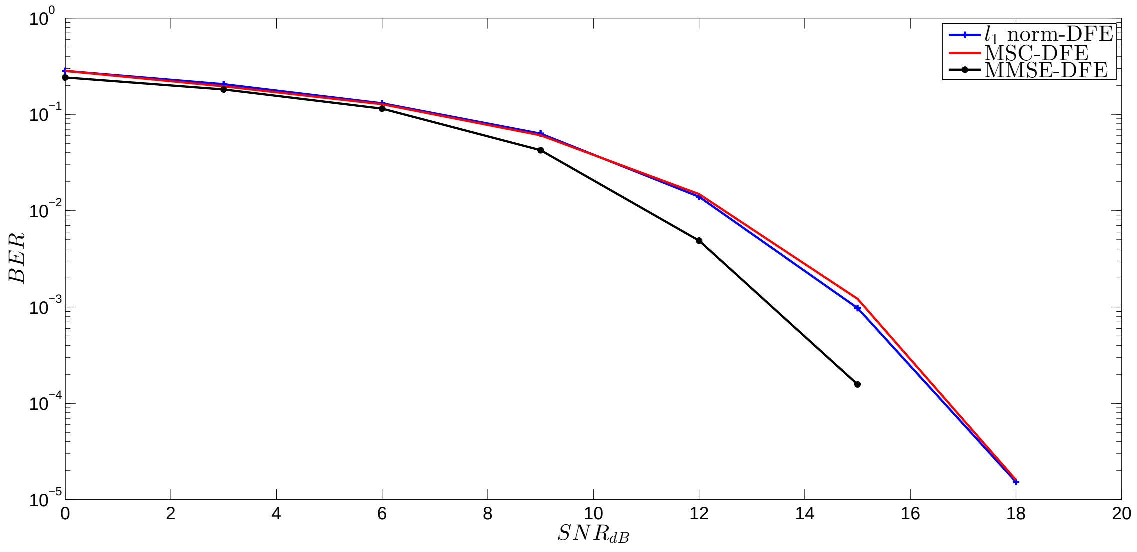


Figure 4.11: Bit error rate for $\epsilon' = 0.4$.

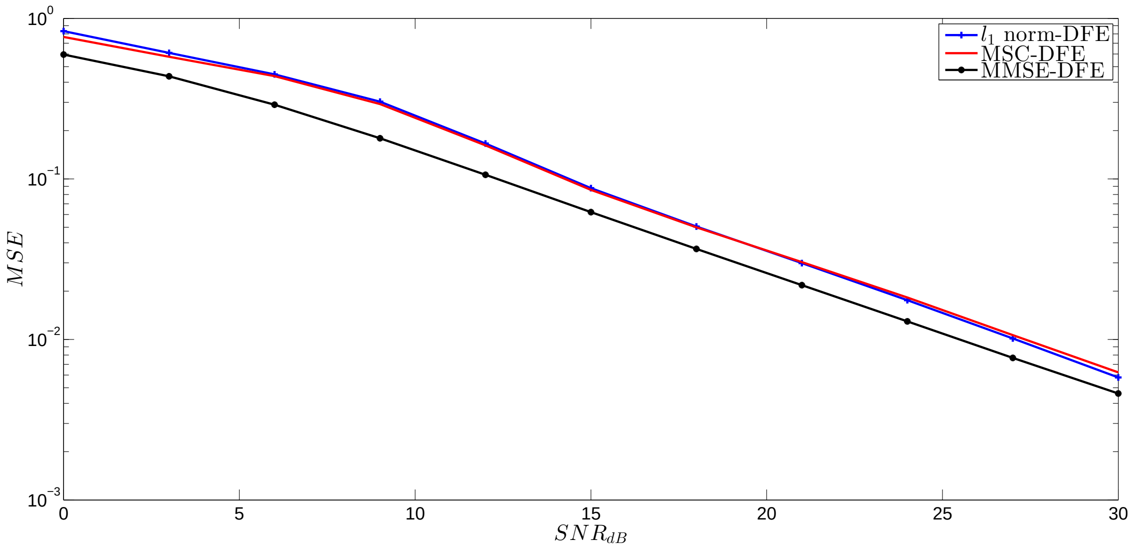


Figure 4.12: Mean square error for $\epsilon' = 0.4$.

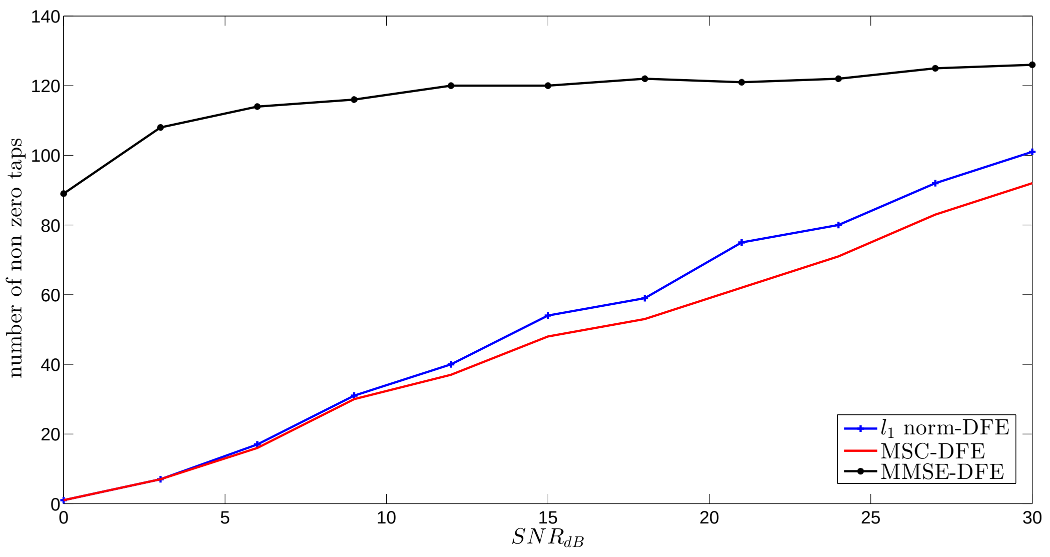


Figure 4.13: Tap support for $\epsilon' = 0.4$.

By adjusting the λ used in the MSC algorithm, we observe that we can still obtain a sparser filer, and also achieve better performance, as shown in Figure 4.14, Figure 4.15 and Figure 4.16.

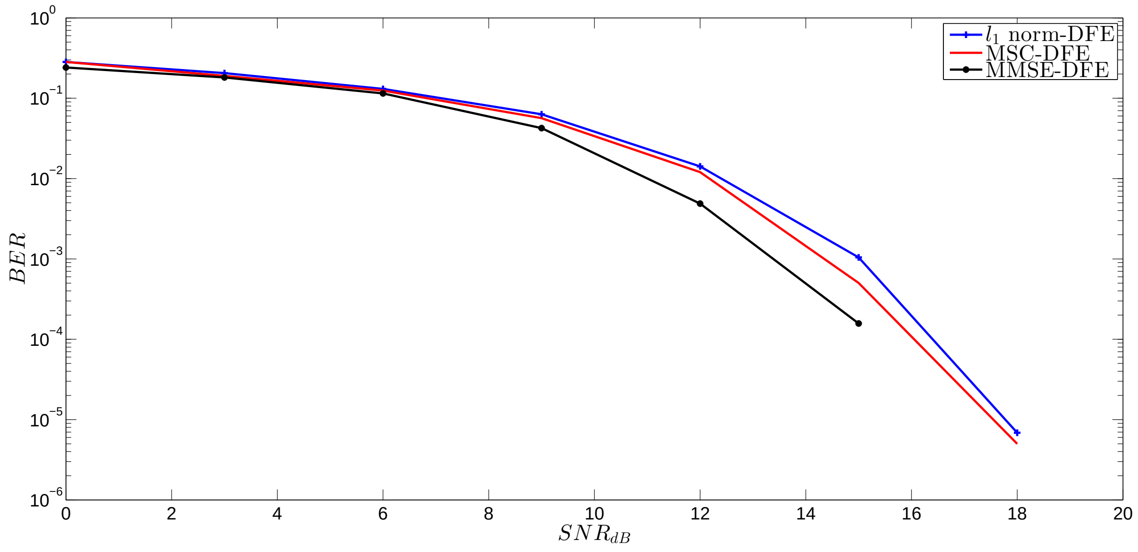


Figure 4.14: Bit error rate for $\epsilon' = 0.4$ and adjusting λ.

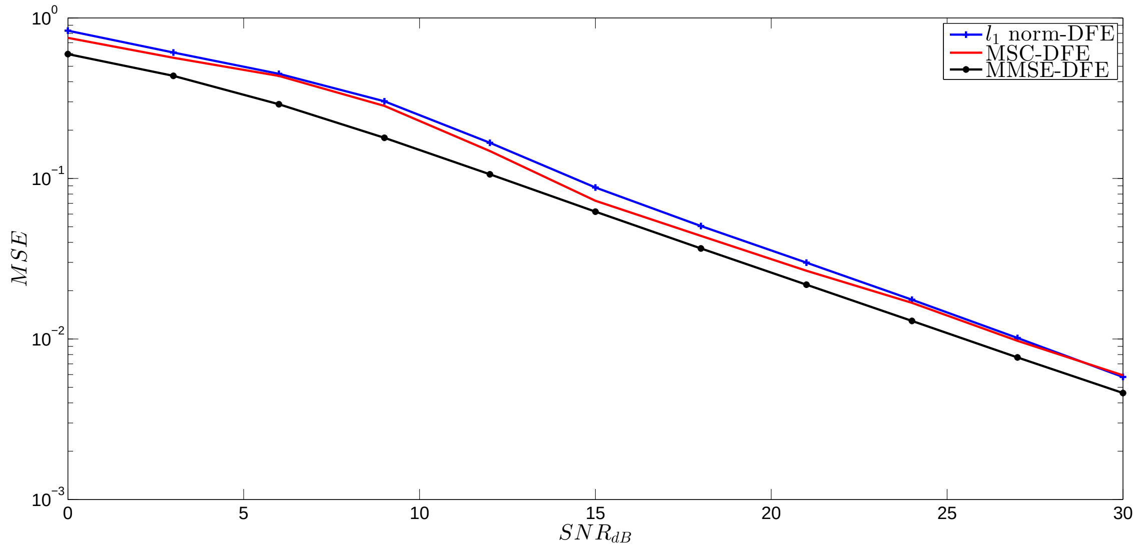


Figure 4.15: Mean square error for $\epsilon' = 0.4$ and adjusting λ.

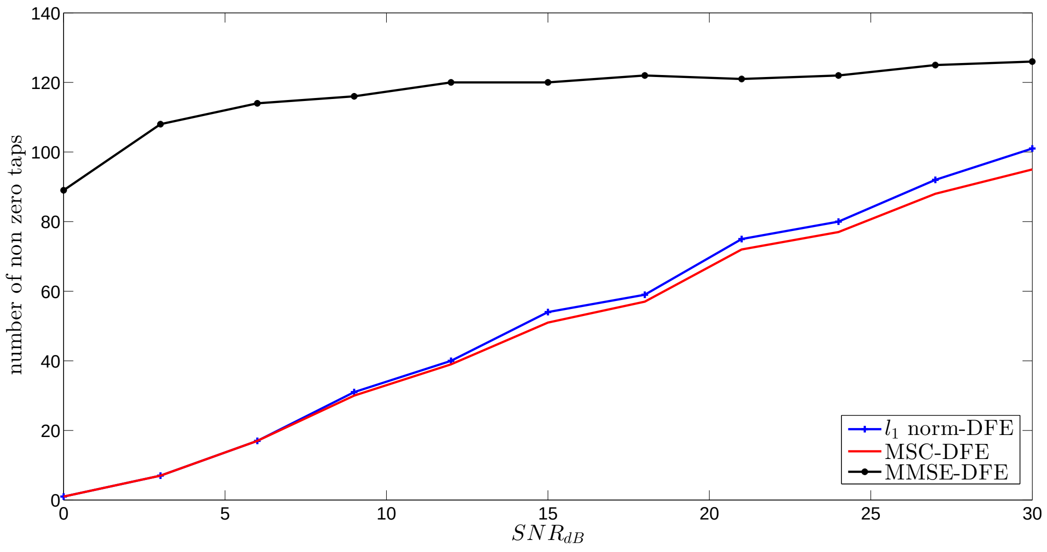


Figure 4.16: Tap support for $\epsilon' = 0.4$ and adjusting λ.

Finally, in order to understand the structure of the sparse filters, we plot in Figure (4.17) the vector stacking the FFF and FBF tap weights of the MMSE-DFE, the l_1 norm DFE and the MSC-DFE, for SNR 15dB. We observe that the sparse filters increase the amplitude of some taps, since they do not have the assistance of small taps in the equalization. This increment is more apparent in the MSC-DFE filters, since they have less non-zero taps.

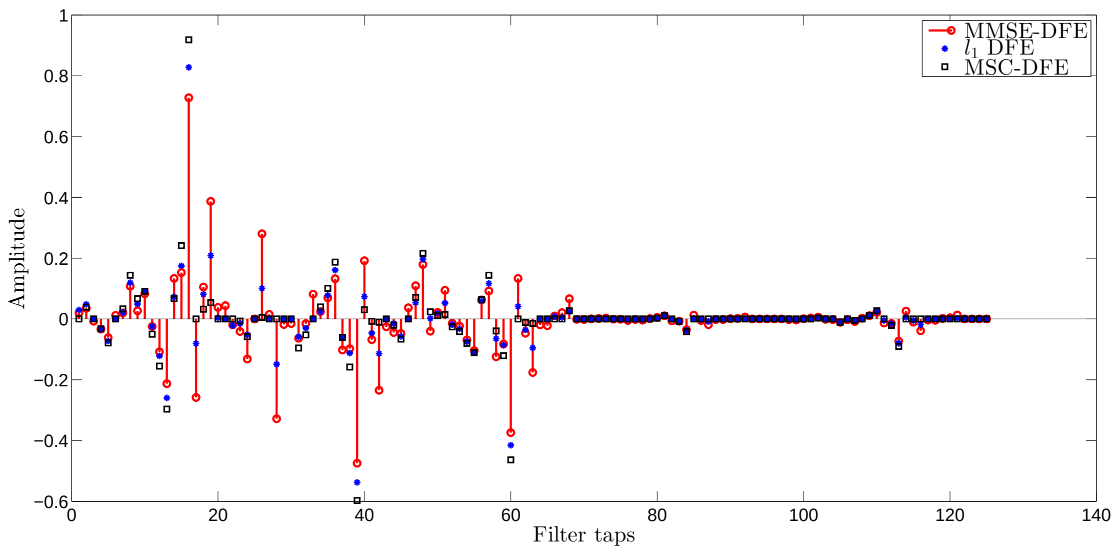


Figure 4.17: DFE filters.

Chapter 5

Conclusion

In this thesis, we considered the estimation and equalization of sparse multipath channels using convex optimization techniques. We examined the MSC algorithm, which solves convex optimization problems using non-convex and more sparsity promoting penalty functions than the l_1 norm. The MSC algorithm was used for the channel estimation problem and provided us with more accurate estimates than the l_1 norm regularized approximation problem. The MSC algorithm was also used for the computation of sparse DFE filters. It was shown that we can obtain more sparse filters and also achieve the same or better performance than when the l_1 norm regularization is used.

Appendix A

The $(N_f + N_b) \times (N_f + N_b)$ auto-correlation matrix appeared in the second term of (4.1) can be analysed as

$$\begin{aligned}
\mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}} &= E[\tilde{\mathbf{r}}\tilde{\mathbf{r}}^H] \\
&= E\left[\begin{bmatrix} \mathbf{r}_{n:n-N_f+1} \\ \mathbf{u}_{n-\Delta-1:n-\Delta-N_b} \end{bmatrix} [\mathbf{r}^H_{n:n-N_f+1} \mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}]\right] \\
&= E\begin{bmatrix} \mathbf{r}_{n:n-N_f+1}\mathbf{r}^H_{n:n-N_f+1} & \mathbf{r}_{n:n-N_f+1}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b} \\ \mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{r}^H_{n:n-N_f+1} & \mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b} \end{bmatrix} \\
&= \begin{bmatrix} E[\mathbf{r}_{n:n-N_f+1}\mathbf{r}^H_{n:n-N_f+1}] & E[\mathbf{r}_{n:n-N_f+1}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] \\ E[\mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{r}^H_{n:n-N_f+1}] & E[\mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] \end{bmatrix}, \qquad (5.1)
\end{aligned}$$

where

$$E[\mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] = \sigma_u^2\ \mathbf{I}_{N_b},$$

and the $(N_f \times N_f)$ output auto-correlation matrix $\mathbf{R}_{\mathbf{rr}}$ can be analysed as

$$\begin{aligned}
\mathbf{R}_{\mathbf{rr}} &= E[\mathbf{r}_{n:n-N_f+1}\mathbf{r}^H_{n:n-N_f+1}] \\
&= E\left[\left(\mathbf{H}\mathbf{u}_{n:n-N_f-L+1} + \mathbf{w}_{n:n-N_f+1}\right)\left(\mathbf{H}\mathbf{u}_{n:n-N_f-L+1} + \mathbf{w}_{n:n-N_f+1}\right)^H\right] \\
&= \mathbf{H}E\left[\mathbf{u}_{n:n-N_f-L+1}\mathbf{u}^H_{n:n-N_f-L+1}\right]\mathbf{H}^H + \mathbf{H}E\left[\mathbf{u}_{n:n-N_f-L+1}\mathbf{w}^H_{n:n-N_f+1}\right] + \\
&\quad E\left[\mathbf{w}_{n:n-N_f+1}\mathbf{u}^H_{n:n-N_f-L+1}\right]\mathbf{H}^H + E\left[\mathbf{w}_{n:n-N_f+1}\mathbf{w}^H_{n:n-N_f+1}\right].
\end{aligned}$$

Since the the data sequence is independent from the noise, the cross-correlation matrices $E\left[\mathbf{u}_{n:n-N_f-L+1}\mathbf{w}^H_{n:n-N_f+1}\right]$ and $E\left[\mathbf{w}_{n:n-N_f+1}\mathbf{u}^H_{n:n-N_f-L+1}\right]$ are equal to $\mathbf{0}$. Also, the $(N_f + L) \times (N_f + L)$ input auto-correlation matrix is given by

$$\mathbf{R}_{\mathbf{uu}} = E[\mathbf{u}_{n:n-N_f-L+1}\mathbf{u}^H_{n:n-N_f-L+1}] = \sigma_u^2\ \mathbf{I}_{N_f+L},$$

and the $(N_f \times N_f)$ noise auto-correlation matrix is given by

$$\mathbf{R}_{ww} = E[\mathbf{w}_{n:n-N_f+1}\mathbf{w}^H_{n:n-N_f+1}] = \sigma_w^2\ \mathbf{I}_{N_f}.$$

Thus, $\mathbf{R}_{\mathbf{rr}}$ is given by

$$\mathbf{R}_{\mathbf{rr}} = \mathbf{H}\mathbf{R}_{\mathbf{uu}}\mathbf{H}^H + \mathbf{R}_{\mathbf{ww}},$$

and (5.1) is rewritten as

$$\mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}} = \begin{bmatrix} \mathbf{R}_{\mathbf{rr}} & E[\mathbf{r}_{n:n-N_f+1}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] \\ E[\mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{r}^H_{n:n-N_f+1}] & \sigma_u^2\mathbf{I}_{N_b} \end{bmatrix}. \tag{5.2}$$

Let $s = N_f + L - N_b - 1$. If $\Delta \leq s$, we have

$$\begin{aligned}
E[\mathbf{r}_{n:n-N_f+1}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] &= \mathbf{H}\ E[\mathbf{u}_{n:n-N_f-L+1}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] \\
&= \mathbf{H}\ E\left[\begin{bmatrix} \mathbf{u}_{n:n-\Delta} \\ \mathbf{u}_{n-\Delta-1:n-\Delta-N_b} \\ \mathbf{u}_{n-\Delta-N_b-1:n-N_f-L+1} \end{bmatrix} \mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}\right] \\
&= \mathbf{H}\ E\begin{bmatrix} \mathbf{u}_{n:n-\Delta}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b} \\ \mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b} \\ \mathbf{u}_{n-\Delta-N_b-1:n-N_f-L+1}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b} \end{bmatrix} \\
&= \mathbf{H}\ \begin{bmatrix} E\ [\mathbf{u}_{n:n-\Delta}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] \\ E\ [\mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] \\ E\ [\mathbf{u}_{n-\Delta-N_b-1:n-N_f-L+1}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] \end{bmatrix} \\
&= \mathbf{H}\ \begin{bmatrix} 0_{(\Delta+1)\times N_b} \\ \sigma_u^2\mathbf{I}_{N_b} \\ 0_{s-\Delta\times N_b} \end{bmatrix} \\
&= \sigma_u^2\mathbf{H}\ \begin{bmatrix} 0_{(\Delta+1)\times N_b} \\ \mathbf{I}_{N_b} \\ 0_{s-\Delta\times N_b} \end{bmatrix}.
\end{aligned}$$

If $\Delta > s$, we have

$$
\begin{aligned}
E[\mathbf{r}_{n:n-N_f+1}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] &= \mathbf{H}\ E[\mathbf{u}_{n:n-N_f-L+1}\mathbf{u}^H_{n-\Delta-1:n-\Delta-N_b}] \\
&= \mathbf{H}\ E\left[\begin{bmatrix} \mathbf{u}_{n:n-\Delta} \\ \mathbf{u}_{n-\Delta-1:n-N_f-L+1} \end{bmatrix} \begin{bmatrix} \mathbf{u}^H_{n-\Delta-1:n-N_f-L+1} & \mathbf{u}^H_{n-N_f-L:n-\Delta-N_b} \end{bmatrix}\right] \\
&= \mathbf{H}\ E\begin{bmatrix} \mathbf{u}_{n:n-\Delta}\mathbf{u}^H_{n-\Delta-1:n-N_f-L+1} & \mathbf{u}_{n:n-\Delta}\mathbf{u}^H_{n-N_f-L:n-\Delta-N_b} \\ \mathbf{u}_{n-\Delta-1:n-N_f-L+1}\mathbf{u}^H_{n-\Delta-1:n-N_f-L+1} & \mathbf{u}_{n-\Delta-1:n-N_f-L+1}\mathbf{u}^H_{n-N_f-L:n-\Delta-N_b} \end{bmatrix} \\
&= \mathbf{H}\ \begin{bmatrix} E[\mathbf{u}_{n:n-\Delta}\mathbf{u}^H_{n-\Delta-1:n-N_f-L+1}] & E[\mathbf{u}_{n:n-\Delta}\mathbf{u}^H_{n-N_f-L:n-\Delta-N_b}] \\ E[\mathbf{u}_{n-\Delta-1:n-N_f-L+1}\mathbf{u}^H_{n-\Delta-1:n-N_f-L+1}] & E[\mathbf{u}_{n-\Delta-1:n-N_f-L+1}\mathbf{u}^H_{n-N_f-L:n-\Delta-N_b} \end{bmatrix} \\
&= H\ \begin{bmatrix} \mathbf{0}_{(\Delta+1)\times(N_f+L-1-\Delta)} & \mathbf{0}_{(\Delta+1)\times(\Delta+N_b-N_f-L+1)} \\ \sigma_u^2\mathbf{I}_{s+N_b-\Delta} & \mathbf{0}_{(s-N_b-\Delta)\times(\Delta-s)} \end{bmatrix} \\
&= \sigma_u^2\mathbf{H}\ \begin{bmatrix} \mathbf{0}_{(\Delta+1)\times N_b} & \\ \mathbf{I}_{s+N_b-\Delta} & \mathbf{0}_{(s-N_b-\Delta)\times(\Delta-s)} \end{bmatrix}.
\end{aligned}
$$

We denote $\mathbf{J}_\Delta$ a $(N_f+L)\times N_b$ matrix, whose structure depends on Δ as follows

$$
\mathbf{J}_\Delta = \begin{cases} \begin{bmatrix} \mathbf{0}_{(\Delta+1)\times N_b} \\ \mathbf{I}_{N_b} \\ \mathbf{0}_{s-\Delta\times N_b} \end{bmatrix}, & \text{for } \Delta \le s, \\ \begin{bmatrix} \mathbf{0}_{(\Delta+1)\times N_b} & \\ \mathbf{I}_{s+N_b-\Delta} & \mathbf{0}_{(s-N_b-\Delta)\times(\Delta-s)} \end{bmatrix}, & \text{for } \Delta > s. \end{cases}
$$

Then (5.1) becomes

$$
\mathbf{R}_{\tilde{\mathbf{r}}\tilde{\mathbf{r}}} = \begin{bmatrix} \mathbf{R}_{\mathbf{rr}} & \sigma_u^2\mathbf{H}\mathbf{J}_\Delta \\ \sigma_u^2\mathbf{J}_\Delta^H\mathbf{H}^H & \sigma_u^2\mathbf{I}_{N_b} \end{bmatrix}.
$$

The cross-correlation appeared in the third term of (4.1), is computed as

$$\begin{aligned}
E[\tilde{\mathbf{r}}\ u^*_{n-\Delta}] &= E[\tilde{\mathbf{r}}\mathbf{u}^H_{n:n-N_f-L+1}\mathbf{e}_\Delta] \\
&= E[\tilde{\mathbf{r}}\mathbf{u}^H_{n:n-N_f-L+1}]\mathbf{e}_\Delta \\
&= E\left[\begin{bmatrix} \mathbf{r}_{n:n-N_f+1} \\ \mathbf{u}_{n-\Delta-1:n-\Delta-N_b} \end{bmatrix} \mathbf{u}^H_{n:n-N_f-L+1}\right]\mathbf{e}_\Delta \\
&= \begin{bmatrix} E[\mathbf{r}_{n:n-N_f+1}\ \mathbf{u}^H_{n:n-N_f-L+1}] \\ E[\mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{u}^H_{n:n-N_f-L+1}] \end{bmatrix}\mathbf{e}_\Delta \\
&= \begin{bmatrix} \mathbf{H}\ E[\mathbf{u}_{n:n-N_f-L+1}\ \mathbf{u}^H_{n:n-N_f-L+1}] \\ E[\mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{u}^H_{n:n-N_f-L+1}] \end{bmatrix}\mathbf{e}_\Delta \\
&= \begin{bmatrix} \sigma_u^2\mathbf{H} \\ E[\mathbf{u}_{n-\Delta-1:n-\Delta-N_b}\mathbf{u}^H_{n:n-N_f-L+1}] \end{bmatrix}\mathbf{e}_\Delta \\
&= \begin{bmatrix} \sigma_u^2\mathbf{H} \\ \sigma_u^2\mathbf{J}_\Delta \end{bmatrix}\mathbf{e}_\Delta
\end{aligned}$$

We denote $\tilde{\mathbf{p}}_\Delta = \begin{bmatrix} \sigma_u^2\mathbf{H} \\ \sigma_u^2\mathbf{J}_\Delta \end{bmatrix}\mathbf{e}_\Delta$.

Bibliography

[1] I. W. Selesnick and I. Bayram, *"Sparse signal estimation by maximally sparse convex optimization,"* IEEE Trans. Signal Process., vol. 62, no. 5, pp. 1078-1092, Mar. 2014.

[2] A. Gomaa and N. Al-Dhahir, *"A new design framework for sparse MIMO equalizers,"* IEEE Trans. Commun., vol. 59, no. 8, pp. 2132-2140, Aug. 2011.

[3] N. Al-Dhahir and J. M. Cioffi, *"MMSE decision-feedback equalizers: finite-length results,"* IEEE Trans. Inf. Theory, vol. 41, no. 4, pp. 961 - 975, July 1995.

[4] G. Gui, Q. Wan, W. Peng, and F. Adachi, *"Sparse multipath channel estimation using compressive sampling matching pursuit algorithm,"* IEEE VTS APWCS '10, Kaohsiung, Taiwan, pp. 10-14, May 2010.

[5] J. Fan and R. Li, *"Variable selection via nonconcave penalized likelihood and its oracle properties,"* J. Am. Stat. Assoc., vol. 96, no. 456, pp. 1348-1359, 2001.

[6] J. Haupt, W. U. Bajwa, G. Raz, and R. Nowak, *"Toeplitz compressed sensing matrices with applications to sparse channel estimation,"* IEEE Trans. Inf. Theory, vol. 56, no. 11, pp. 5862-5875, Nov. 2010.

[7] F. K. H. Lee, *"A study of two equalization techniques for sparse multipath channels,"* Ph.D. dissertation, Dept. Elect. Eng., Queen's Univ., Kingston, Ontario, Canada, 2003.

[8] J. Mattingley and S. Boyd, *"Real-time convex optimization in signal processing,"* IEEE Signal Process. Mag., vol. 27, no. 3, pp. 50-61, May 2010.

[9] R. Tibshirani, *"Regression shrinkage and selection via the lasso,"* J. Royal. Statist. Soc B., vol. 58, no. 1, pp. 267-288, 1996.

[10] S. Chen, D. Donoho and M. Saunders, *"Atomic decomposition by basis pursuit,"* SIAM J. on Sci. Comp., vol. 20, no. 1, pp. 33-61, 1998.

[11] F. Bach, R. Jenatton, J. Mairal and G. Obozinski. *"Optimization with sparsity-inducing penalties,"* Foundations and Trends in Machine Learning, vol. 4, no. 1, pp. 1-106, 2012.

[12] A. P. Liavas ,*“Telecommunication Systems II Lecture Notes,”* 2010.

[13] S. Boyd and L. Vandenberghe, *“Convex optimization,”* Cambridge University Press, 2004.

[14] J. G. Proakis and M. Salehi, *“Digital communications,”* McGraw-Hill, 1995.

[15] A. Goldsmith, *“Wireless communications,”* Cambridge University Press, 2005.

[16] C. D. Meyer, *“Matrix analysis and applied linear algebra,”* SIAM, 2000.