A STUDY OF LMS-BASED ALGORITHMS: EXPLOITING PLAIN AND HIDDEN SPARSITY

Gabriel da Silva Chaves

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientadores: Markus Vinícius Santos Lima
Tadeu Nagashima Ferreira

Rio de Janeiro
Agosto de 2019

A STUDY OF LMS-BASED ALGORITHMS: EXPLOITING PLAIN AND HIDDEN SPARSITY

Gabriel da Silva Chaves

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Examinada por:

Prof. Markus Vinícius Santos Lima, D.Sc.

Prof. Tadeu Nagashima Ferreira, D.Sc.

Prof. Diego Barreto Haddad, D.Sc.

Prof. Kenji Nose Filho, D.Sc.

RIO DE JANEIRO, RJ – BRASIL
AGOSTO DE 2019

Chaves, Gabriel da Silva

A Study of LMS-based Algorithms: Exploiting Plain and Hidden Sparsity/Gabriel da Silva Chaves. – Rio de Janeiro: UFRJ/COPPE, 2019.

XII, 64 p.: il.; 29, 7cm.

Orientadores: Markus Vinícius Santos Lima
Tadeu Nagashima Ferreira

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2019.

Referências Bibliográficas: p. 57 – 64.

1. adaptive filtering. 2. sparsity. 3. LMS-based algorithms. 4. proportionate-type family. 5. feature matrix. 6. discard function. I. Lima, Markus Vinícius Santos *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

À minha família.

Agradecimentos

Agradeço aos meu pais, por todo apoio e carinho que me deram durante a conclusão dessa jornada, e por todos os sacrifícios que fizeram, para que essa etapa da minha vida fosse a mais agradável possível.

Agradeço ao meu irmão Rafael, melhor pessoa que conheço, por estar sempre comigo, ao meu lado e me apoiando em todas as situações imagináveis. Com certeza, sem você eu não estaria onde estou hoje, e sei que sempre poderei contar com sua ajuda.

Agradeço aos meus amigos, por me auxiliarem durante essa jornada. Sem vocês eu não teria conseguido: João Pedro, Wescley, Letícia, Roberta, Thiago e Mariana. Em especial, agradeço aqueles que estavam mais próximos durante esses últimos meses, e sempre dispostos a ouvir minhas reclamações: Rômulo, Simone, Mateus, João Santos, João Vicente, Roberto, Vinícius, Wesley, Matheus, Lucas Cinelli, Igor, Felipe e Marcelo. Agradeço aos meus antigos amigos, que mesmo longe, foram muito importantes durante todo esse tempo: Hugo, Leon e Lucas Souza.

Agradeço aos meus orientadores: Markus e Tadeu, que para mim são muito mais que professores. Obrigado pela confiança que tiveram em mim, por todo conhecimento que me foi passado e por todas oportunidades oferecidas. Obrigado também pela paciência que tiveram comigo. Vou levar para a vida tudo que me ensinaram. Vocês contribuíram para o meu crescimento profissional e pessoal, sou grato por tudo.

Agradeço aos professores Diego Barreto Haddad e Kenji Nose Filho, por aceitarem o convite para compor a banca avaliadora deste trabalho.

Por fim, agradeço à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio financeiro (Código de Financiamento 001).

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM ESTUDO DE ALGORITMOS BASEADOS EM LMS: EXPLORANDO ESPARSIDADE SIMPLES E ESCONDIDA

Gabriel da Silva Chaves

Agosto/2019

Orientadores: Markus Vinícius Santos Lima
Tadeu Nagashima Ferreira

Programa: Engenharia Elétrica

Nas últimas décadas, os filtros adaptativos foram empregados em diferente aplicações. O foco dessa dissertação é em uma aplicação em particular: identificação de sistemas que apresentam algum esparsidade. Um sistema é esparso quando ele pode ser representado por poucos coeficientes em algum domínio. Este trabalho aborda os seguintes tipos de esparsidade: a simples e a escondida. A simples é bastante conhecida em filtragem adaptativa e ocorre quando é possível observar a esparsidade sem nenhuma manipulação matemática. Por conta da sua importância, duas famílias de algoritmos foram criadas com o intuito de explorará-la. A ideia proporcional tenta explorar a esparsidade ao atribuir tamanho de passo únicos para cada coeficiente. Enquanto a família dos algoritmos regularizados usam funções promotoras de esparsidade. Considerando que os algoritmos proporcionais não modelam a esparsidade explicitamente, esse trabalho realiza um estudo aprofundado dessa família, apresentando algumas propriedades ainda não documentadas. Para isso, resultados numéricos foram apresentados, concluindo-se que os algoritmos proporcionais exploram algo mais geral que a esparsidade. A esparsidade escondida é mais recente em filtragem adaptativa e requer manipulações matemáticas para revelar a esparsidade do sistema. O algoritmo *feature least-mean-square* (F-LMS) resolve esse problema ao introduzir a matriz de *features*. Entretanto, existem muitas aplicações que possuem os dois tipos de esparsidade, e o algoritmo F-LMS não explora a simples. Com isso, foi proposto o algoritmo *simple sparsity-aware* F-LMS (SSF-LMS), o qual explora ambos os tipos de esparsidade enquanto realiza menos operações aritméticas, ao usar a função de discarte.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A STUDY OF LMS-BASED ALGORITHMS: EXPLOITING PLAIN AND HIDDEN SPARSITY

Gabriel da Silva Chaves

August/2019

Advisors: Markus Vinícius Santos Lima
Tadeu Nagashima Ferreira

Department: Electrical Engineering

Over the last decades, adaptive filters have been used in many different applications. This dissertation focuses on one in particular: the identification of systems presenting some kind of sparsity. A system is sparse when we represent it by only a few coefficients in some domain. This work addresses the following kinds of sparsity: the plain and the hidden. The plain one is well-known in the adaptive filtering field and occurs when the sparsity is directly observed in the coefficients space, without any mathematical manipulation. There are two distinct families of algorithms that were intended to tackle this problem. The proportionate family tries to exploit the plain sparsity by assigning particular step sizes to each of the coefficients. The second family is the regularized-type, which relies on sparsity promoting functions. Since the proportionate family of algorithms does not model the sparsity explicitly, this work performs a thorough study of this family, addressing some undocumented properties. Therefore, we present several numerical results, concluding that they exploit something more general than the plain sparsity. The hidden sparsity is a more recent problem in the adaptive filtering field, in which we reveal the system sparsity through some mathematical manipulation. The feature least-mean-square (F-LMS) algorithm addresses this problem introducing a feature matrix in the objective function of the LMS algorithm. However, several applications have both types of sparsity, and the F-LMS can not exploit the plain one. Then, we propose the simple sparsity-aware F-LMS (SSF-LMS), which exploits both of them while requiring less arithmetic operations by using the discard function.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A digital filter is a structure that extracts the desired information about a prescribed quantity of interest from a set of noisy data. In other words, it maps an input signal, which contains the desired information, to an output signal that facilitates the visualization of such information [1–4]. We can separate a filter into two classes: linear and nonlinear. When the output signal is a linear combination of the system input, we have a linear filter. Otherwise, it is nonlinear [1–4]. Many filters are time-invariant, meaning that their internal coefficients are fixed, designed according to prescribed specifications. However, sometimes, a time-invariant filter cannot satisfy the specifications, either because they are time-varying or unknown [1].

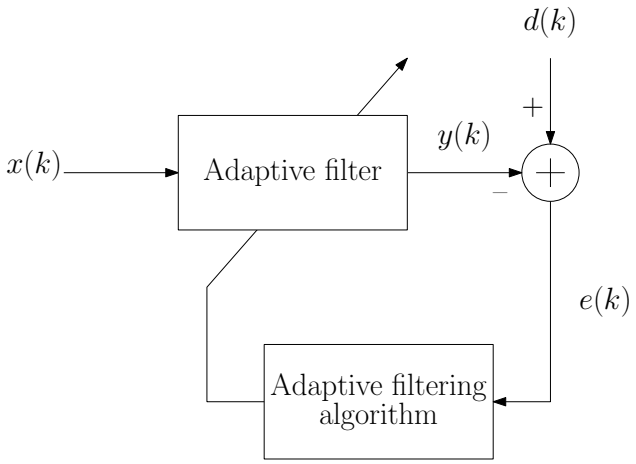


Figure 1.1: General representation of an adaptive filter.

When a time-invariant filter cannot satisfy the prescribed specification, we can use an adaptive filter [1, 5, 6]. As they depend on the input signal, adaptive filters are nonlinear systems. Figure 1.1 depicts the general representation of an adaptive filter, where k is the iteration number, $x(k)$ is the input signal, $y(k)$ denotes the output of the adaptive filter, and $d(k)$ specifies the desired signal. The error signal is $e(k) = d(k) - y(k)$. We use it to form an objective function that determines the appropriate updating of the filter coefficients for an adaptive filtering algorithm.

The minimization of the objective function indicates that the output and the desired signal are matching in some sense [1].

The main goal of an adaptive filter is to adjust its coefficients in order to minimize a given objective function [1]. Generally, this function F depends on the input, desired and output signals [1, 3]. A widely used objective function is the mean-square error (MSE) given by

$$F[e(k)] = \mathbb{E}\left[|e(k)|^2\right] \ , \tag{1.1}$$

where $\mathbb{E}[.]$ denotes expected value.

The solution for the minimization of (1.1) is the well known Wiener filter, which is the optimum solution for linear filters in terms of the MSE criterion [1, 7, 8]. However, this solution is only theoretical, since in order to design the Wiener filter we need an infinite amount of information to calculate some statistical parameters (mean and correlation) [9–11]. Instead of the MSE, one can use the instantaneous square error (ISE) objective function given by

$$F[e(k)] = |e(k)|^2 \ . \tag{1.2}$$

The procedure that minimizes the objective function is the adaptive filtering algorithm, depicted in Figure 1.1. In particular, one of the most popular is the least-mean-square (LMS) algorithm. Since its development [12], it constitutes a benchmark among other adaptive filtering algorithms [13]. Unlike the Wiener solution, the LMS algorithm minimizes the ISE, instead of the MSE. By using this objective function, the LMS algorithm does not require to compute the correlation and cross-correlation matrices, which are necessary to calculate the Wiener solution [1, 3]. The LMS algorithm relies on these simple approaches to save computational resources. Despite its simplicity, it achieves satisfactory performance [3]. The normalized LMS (NLMS) algorithm is the natural upgrade of the LMS algorithm [1, 3, 6]. It addresses some of the problems present in the LMS algorithm by normalizing the input signal, allowing the NLMS algorithm to reach faster convergence speed without using estimates of the input signal correlation matrix [1]. One can interpret the normalization of the input signal as a variable convergence factor, [1] which is a trivial approach to improve the minimization of the instantaneous output error [1]. Thus, the NLMS algorithm is a faster and more stable version of the LMS algorithm that usually outperforms it [1, 3, 6]. However, to achieve all these improvements, the NLMS algorithm requires more computational complexity. Although the focus of this work is on the system identification, we can

[1] In this text, we use both convergence factor and step size with the same meaning.

employ the LMS-based algorithms in several practical applications, such as channel equalization [14–18], active noise control [19–22], continuous time-filter tuning [23], and system identification [1, 24, 25], among others [26].

There are a plethora of practical systems that involve sparse impulse responses, such as echo path channels [27], wireless channels [28], 5G wireless systems [29], medical data [30], channel equalization [31], seismic data [32], preamplifiers [33], among others [34–36]. The sparsity in a system ensues when just a few of its coefficients are relevant, i.e., the impulse response has few nonzero coefficients [27, 34, 37]. In the adaptive filtering field, several research efforts have been made to address the *plain* sparsity problem, which appears when the impulse response has a sparse representation without any mathematical manipulation, depicted in Figure 1.2. Many algorithms have been developed with the purpose to exploit the plain sparsity in the system. The proportionate NLMS (PNLMS) algorithm gives birth to an entire family of algorithms that aims at the exploitation of the sparsity, the so-called proportionate-type family [27]. These algorithms assign particular step sizes to each coefficient, proportional to their magnitude. Several important algorithms compose this family, such as the PNLMS++ [38], improved PNLMS (IPNLMS) [39], μ-law PNLMS (MPNLMS) [40], improved MPNLMS (IMPNLMS) [41], proportionate affine-projection (PAP) [42], and improved PAP (IPAP) [43] algorithms, and many others [44–47].

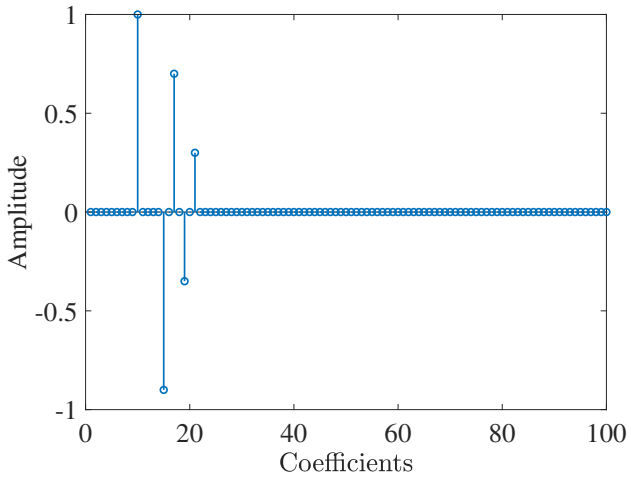
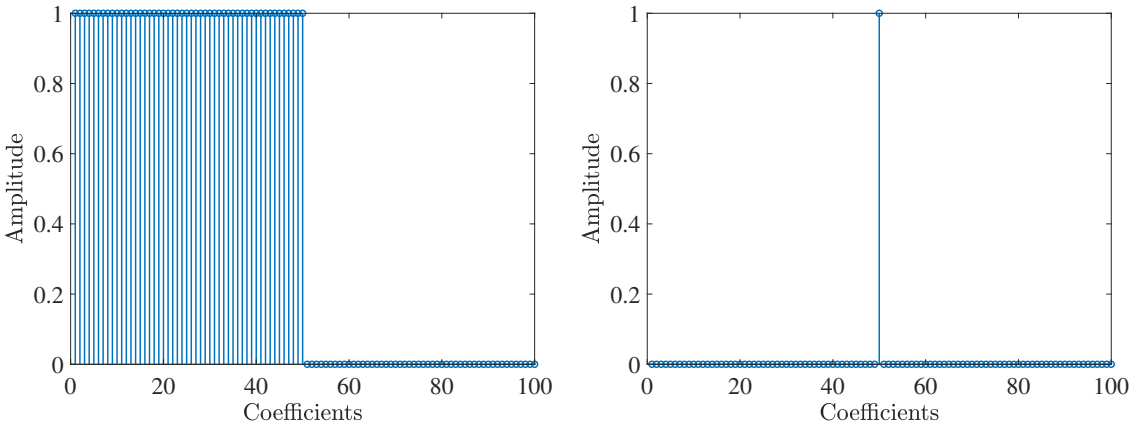


Figure 1.2: Plain sparse impulse response.

Recently, the regularized family has emerged, which can exploit the system sparsity, explicitly. The algorithms of this family use penalty functions to promote sparsity. We can best represent a sparse vector by its ℓ_0-"norm", which counts the

number of nonzero entries present in the vector. [2] Thus, these algorithms impose a penalization by the ℓ_0-norm in the adaptive filter coefficients, changing the objective function that the algorithm minimizes. However, by using the ℓ_0-norm, we have a combinatorial optimization problem, which is hard to solve and has high computational complexity [50]. Then, it is common to use an approximation of the ℓ_0-norm or the ℓ_1-norm [50–52]. Several algorithms compose this family, as the ℓ_0-LMS [53], zero-attracting LMS (ZA-LMS) [54], and reweighted ZA-LMS (RZA-LMS) [54] algorithms.



(a) In the coefficients domain (current representation).

(b) In the transformed domain.

Figure 1.3: Hidden sparse impulse response.

There is another type of sparsity besides the plain one. We refer to *hidden* sparsity when some mathematical manipulation in the coefficients is required to reveal the system sparsity. In other words, we cannot observe the sparsity in the current representation of the system. Figure 1.3 depicts an impulse response with hidden sparsity and its equivalent in a transformed domain. In Figure 1.3a, one can notice the impulse response in the coefficients domain, which has the hidden sparsity. By performing mathematical manipulations, we have the sparse impulse response in a transformed domain, depicted in Figure 1.3b. Unfortunately, none of the former adaptive filtering algorithms can exploit this type of sparsity. However, the new feature LMS (F-LMS) algorithm benefits from the hidden sparsity by incorporating some features implicit to the unknown system to its objective function. These features can refer to any characteristics of the system, such as the band of the spectrum or decomposition of the impulse response [13, 55]. However, many systems have both types of sparsity, and the F-LMS algorithm does not take advantage of the plain one. Then, we propose a low computational complexity algorithm that

[2] We represent the ℓ_0-"norm" with the quotation marks because it is not a proper norm. The ℓ_0 does not satisfy the property of homogeneity, i.e., it is a function that does not satisfy all the conditions to be a norm [48]. Actually, the ℓ_0 is a quasi-norm [49]. However, for simplicity, we use ℓ_0-norm (without the quotation marks).

can exploit both sparsities, simultaneously. By also using the so-called discard function [56], the simple sparsity-aware F-LMS (SSF-LMS) algorithm outperforms the F-LMS algorithm when the system has plain and hidden sparsity, requiring fewer arithmetics operations. It is worth mentioning that [57] proposes an alternative approach to exploit both types of sparsity that uses the ℓ_0-norm, but in such work, the number of arithmetic operations required by the proposed algorithm is much larger in comparison with the SSF-LMS algorithm.

1.1 Main Contributions of this Work

This work investigates the properties of sparsity-aware algorithms, mainly LMS-based ones. Throughout the text, we clarify some misunderstandings about the proportionate-type family. We present several simulations to demonstrate that the algorithms of this family do not exploit the system sparsity at all. In fact, they take advantage of some other features, which usually exist in sparse systems. Thus, we introduce some properties of these algorithms aiming at explaining the observed behavior. It is worth pointing out that some of these properties have never been published before and researchers are not aware of them (nor the limitations they imply). We also explore the hidden sparsity problem by proposing a low computational complexity algorithm, which can exploit both types of sparsity, simultaneously. The proposed SSF-LMS algorithm can outperform the F-LMS algorithm when the impulse response has both types of sparsity, doing fewer arithmetic operations. Moreover, we show how one can choose an adequate feature matrix for a couple of simple impulse responses, which is a requirement to exploit the hidden sparsity.

1.2 Organization of the Text

In Chapter 2, we do a brief review of the LMS, NLMS, PNLMS, IPNLMS, ℓ_0-LMS, and ℓ_0-NLMS algorithms, which we use throughout the experiments. We also present their objective functions and summarize their operations, providing an easy way to implement them. Chapter 3 addresses some properties of the proportionate-type algorithms, showing several simulation results to validate them. Also, by further analysis of these algorithms, we can conclude that they do not exploit the sparsity in the system. In Chapter 4, we introduce the hidden sparsity problem, presenting an algorithm that is capable of exploiting this feature. We also propose the SSF-LMS algorithm, which is a low complexity option that exploits both types of sparsity, simultaneously. Finally, we conclude this work in Chapter 5.

1.3 Notation

Scalars are represented by lower-case letters. Vectors (matrices) are denoted by lowercase (uppercase) boldface letters. For a given iteration k, the weight vector and the input vector are denoted by $\mathbf{w}(k)$, $\mathbf{x}(k) = [x(k)\ x(k-1)\ \ldots\ x(k-N+1)]^{\mathrm{T}} \in \mathbb{R}^{\mathrm{N}+1}$, respectively, where N is the adaptive filter order. The optimum system coefficient is denoted by $\mathbf{w}_{\mathrm{o}}$. The error signal at the kth iteration is defined as $e(k) \triangleq d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$, where $d(k) \in \mathbb{R}$ is the desired signal. The ℓ_1-norm of a vector $\mathbf{w} \in \mathbb{R}^{\mathrm{N}+1}$ is given by $\|\mathbf{w}\|_1 = \sum_{i=0}^{\mathrm{N}} |w_i|$. The maximum function of a vector $\mathbf{w} \in \mathbb{R}^{\mathrm{N}+1}$ is $\max\{\mathbf{w}\} = w_i$ if $w_i \geq w_j\ \forall\ i \neq j = 0,\ \ldots\ ,\mathrm{N}$. The diagonal function of a vector $\mathbf{w} \in \mathbb{R}^{\mathrm{N}+1}$ is $\mathsf{Diag}\{\mathbf{w}\} \in \mathbb{R}^{(\mathrm{N}+1)\times(\mathrm{N}+1)}$, i.e., a diagonal matrix with vector $\mathbf{w}$ in its diagonal.

1.4 Publications

This section lists the accepted works relative to most of Chapter 3 and Chapter 4, respectively:

- CHAVES, G. S., LIMA, M. V. S., FERREIRA, T. N. "O Algoritmo PNLMS Realmente Explora a Esparsidade dos Coeficientes?" In: *XXXVII Brazilian Symposium on Telecommunications and Signal Processing*, pp.1-5, Petrópolis, Brazil, September 2019.

- CHAVES, G. S., LIMA, M. V. S., YAZDANPANAH, H., et al. "A Simple Sparsity-aware Feature LMS Algorithm." In: *27th European Signal Processing Conference*, pp.1-5, A Coruña, Spain, September 2019.

Also, we are preparing a journal paper containing the full contribution of Chapter 3.

Chapter 2

Some LMS-based Algorithms

In this chapter, we present a brief description of some algorithms that have been considered in the next chapters. We overview important properties of the least-mean-square (LMS), normalized LMS (NLMS), proportionate NLMS (PNLMS), improved PNLMS (IPNLMS), ℓ_0-LMS, and ℓ_0-NLMS algorithms. For each algorithm, we deduce its objective function and update equation. Moreover, we summarize the algorithms steps in a friendly manner in order to facilitate their implementation.

2.1 The LMS Algorithm

The LMS adaptive algorithm filter is based on the steepest descent method, also referred to as the stochastic gradient method. This type of algorithm searches for the minimum point of a given objective function by following the opposite direction of the gradient vector. The steepest descent method requires a set of initial conditions for the weights and, through iterative updates, the process will converge to a minimum point if the changes in the weights reduce the objective function [5]. The update equation for this method assumes the general form

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \mu \mathbf{g}_{\mathbf{w}}(k) \ , \tag{2.1}$$

where $\mu \in \mathbb{R}_+$ is the step size and $\mathbf{g}_{\mathbf{w}}(k)$ is the gradient of the objective function with response to $\mathbf{w}(k)$.

In the case of an adaptive process that tries to find an exact or approximate solution for the minimum MSE, the gradient vector is $\mathbf{g}_{\mathbf{w}}(k) = \dfrac{\partial \mathbb{E}[e^2(k)]}{\partial \mathbf{w}}$. However, this method requires exponentially more arithmetical operations (mainly multiplications) as the order of the system increases [3]. Also, computing $\mathbf{g}_{\mathbf{w}}(k)$ requires statistical information about the involved signals, which are not usually available [1, 3, 5, 12]. Therefore, the LMS algorithm performs an estimate of the vector $\mathbf{g}_{\mathbf{w}}(k)$, resulting

in $\hat{\mathbf{g}}_{\mathbf{w}}(k) = \dfrac{\partial e^2(k)}{\partial \mathbf{w}}$. By suppressing the expectation operator, only the instantaneous measures are required, making this a low computational complexity solution. Several works have studied the convergence of the LMS algorithm [1, 3, 5, 12, 58].

In order to find the update equation of the LMS algorithm, we must minimize the cost function

$$\xi(k) = e^2(k) \ , \tag{2.2}$$

where $e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$. By doing so, we obtain the gradient of the cost function, resulting in the update equation of the LMS algorithm given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu e(k)\mathbf{x}(k) \ , \tag{2.3}$$

where $\mu \in \mathbb{R}_+$ is a convergence factor or the step size, which should be chosen in a range to guarantee convergence [1]. The Algorithm 1 summarizes the operations required by the LMS algorithm.

Algorithm 1: The LMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0 \ 0 \ \ldots \ 0]^{\mathrm{T}}$

choose a suitable μ

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$

$\mathbf{w}(k+1) = \mathbf{w}(k) + 2\mu\mathbf{x}(k)e(k)$

2.2 The NLMS Algorithm

Choosing an adequate step size for the LMS algorithm may be a challenging task [1, 3, 5, 12]. One natural solution for this problem is to normalize the input signal, which one can interpret as the inclusion of a variable convergence factor, i.e., the algorithm updates with a different step size in each iteration. Then, in exchange of increased computational complexity, the NLMS algorithm usually achieves faster convergence rate than the LMS algorithm [1].

We have the following optimization problem

$$\begin{aligned} &\min \ \|\mathbf{w}(k+1) - \mathbf{w}(k)\|_2^2 \\ &\text{s.t. } d(k) - \mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k+1) = 0. \end{aligned} \tag{2.4}$$

By solving (2.4), we have the following update equation

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mathbf{x}(k)e(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k) + \delta} \ , \tag{2.5}$$

where $\delta \in \mathbb{R}_+$ is the regularization factor, responsible to avoid numerical errors when $\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k)$ becomes small, i.e., divisions by zero. The term $\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k) + \delta$ at the divisor on the right side of (2.5) represents the variable step size. However, we introduce another step size μ to (2.5), aiming at more freedom in the convergence speed of the algorithm. Then, the update equation of the NLMS algorithm is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu \mathbf{x}(k) e(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k) + \delta} \ , \tag{2.6}$$

where $\mu \in (0, 1]$ is the step size. The complete operation of the NLMS algorithm is given in Algorithm 2.

Algorithm 2: The NLMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0 \ 0 \ \ldots \ 0]^{\mathrm{T}}$

choose μ in the range $0 < \mu \leq 1$

$\delta =$ small constant

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$

$\mathbf{w}(k+1) = \mathbf{w}(k) + \dfrac{\mu \mathbf{x}(k) e(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k) + \delta}$

2.3 The PNLMS Algorithm

The plain sparsity is well known in the context of adaptive filters and several research efforts have been done over the years to tackle this type of sparsity [27, 38–41, 53, 54, 59, 60]. An impulse response is said to be sparse (or plain sparse) when most of its energy is concentrated in a few coefficients, while the others have null or almost null contribution, i.e., the majority of coefficients are equal or close to zero and the sparsity is directly observed in the impulse response, without requiring any mathematical transformation [61]. A few decades ago, the PNLMS algorithm was created with the purpose of exploiting this type of sparsity in echo cancellation systems [27]. Generally, this algorithm reduces the overall convergence time in these scenarios, reaching superior convergence speed than the NLMS algorithm [27, 40].

The proportionate-type family assigns variable step sizes for each coefficient of the adaptive filter. These step sizes are proportional to the magnitude of the respective coefficients. For example, large coefficients receive large step sizes [27, 39, 40]. The PNLMS was the first and the simplest algorithm of this family, but it has all the essence of a proportionate algorithm. In order to obtain the update equation of

the PNLMS algorithm, first we need to solve the following optimization problem

$$\begin{aligned} &\min \ \|\mathbf{w}(k+1)-\mathbf{w}(k)\|^2_{\mathbf{G}^{-1}(k)} \\ &\text{s.t. } d(k)-\mathbf{x}^{\mathrm{T}}(k)\mathbf{w}(k+1)=0 \ , \end{aligned} \tag{2.7}$$

where $\|\mathbf{w}(k+1)-\mathbf{w}(k)\|^2_{\mathbf{G}^{-1}(k)} = [\mathbf{w}(k+1)-\mathbf{w}(k)]^{\mathrm{T}}\mathbf{G}^{-1}(k)[\mathbf{w}(k+1)-\mathbf{w}(k)]$ is a norm induced by the matrix $\mathbf{G}^{-1}(k)$.

In (2.7), a norm induced by the matrix $\mathbf{G}^{-1}(k)$ is introduced, differently from (2.4). This matrix $\mathbf{G}(k)$, widely known as the proportionate matrix, is responsible for reweighting the step sizes given to each coefficient. By solving (2.7), [1] the update equation of the PNLMS algorithm is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu\mathbf{G}(k)\mathbf{x}(k)e(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k)+\delta} \ , \tag{2.8}$$

where μ and δ are the same as in (2.6), i.e., the convergence and regularization factor, respectively. The matrix $\mathbf{G}(k)$ is diagonal with each element defined as

$$g_i(k) \triangleq \frac{\gamma_i(k)}{\sum_{n=0}^{\mathrm{N}}\gamma_n(k)}, \ \forall i = \{0,\ldots,\mathrm{N}\} \ , \tag{2.9}$$

where

$$\gamma_i(k) \triangleq \max\left(\rho\left\{\max\left[\delta_p, |w_0(k)|, ..., |w_{\mathrm{N}}(k)|\right]\right\}, |w_i(k)|\right) \ , \tag{2.10}$$

$\rho \in (0,1]$ is a constant to avoid erroneous results when any coefficient goes to zero and $\delta_p \in \mathbb{R}_+$ is to avoid this same type of errors, when all coefficients go to zero [2]. Typical values to ρ and δ are $5/N$ and 0.01, respectively [39]. It is worthy mentioning that small values of ρ lead to a more proportional approach, where the real magnitude of the low-magnitude coefficients are used in (2.10).

The proportionate matrix $\mathbf{G}(k)$ is common to all algorithms in the proportionate-type family. The main difference among the algorithms of this family is the definition of the variables g_i, i.e., each relation of proportionality defines an algorithm of the proportionate-type family. Thus, the PNLMS algorithm is defined by (2.9)

[1] We introduce the step size μ to the solution of (2.7) for the same reasons as in Section 2.3.
[2] Both of these are common problems in the initialization period of the algorithm.

and (2.10). The Algorithm 3 provides a brief description of the PNLMS algorithm.

Algorithm 3: The PNLMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose μ in the range $0 < \mu \leq 1$

choose ρ in the range $0 < \rho \leq 1$

$\delta = \delta_p =$ small constant

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$

Do for $i \leq \mathrm{N}$

$$\gamma_i(k) = \max\left\{\rho \max\left[\delta_p, |w_0(k)|, ..., |w_{\mathrm{N}}(k)|\right], |w_i(k)|\right\}$$

$$g_i(k) = \frac{\gamma_i(k)}{\sum_{n=0}^{\mathrm{N}} \gamma_n(k)}$$

end for

Compute $\mathbf{G} = \mathsf{Diag}\{g_0\ \ldots\ g_{\mathrm{N}}\}$

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu \mathbf{G}(k)\mathbf{x}(k)e(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k) + \delta}$$

2.4 The IPNLMS Algorithm

The robustness of an adaptive filter is desirable, i.e., it needs to have a minimal performance in any situation or scenario. Several research works have reported the poor performance of the PNLMS algorithm when applied to dispersive unknown impulse responses [38–40]. This can be a problem when information about the unknown system is not known *a priori* or when the unknown system is time-varying.

Unlike sparse systems, dispersive impulse responses do not have the majority of its coefficients with magnitude equal to zero, nor large differences in its coefficients magnitudes. Due to this reason, the PNLMS algorithm can not assign proper step sizes to the estimated filter coefficients. A reliable solution for this problem is the IPNLMS algorithm, which introduces an *NLMS* term to (2.10). Apart from the performance improvement in dispersive systems, the IPNLMS algorithm solves the problem of the *maximum function* presented in (2.10). Therefore, the band-aid parameters ρ and δ_p can be removed [39]. The optimization problem becomes the same as for the PNLMS algorithm, but the entries of $\mathbf{G}(k)$ are calculated differently. They are now defined as

$$g_i(k) \triangleq \frac{\gamma_i(k)}{\|\boldsymbol{\gamma}(k)\|_1}\ ,\ \forall i = \{0, \ldots, \mathrm{N}\}\ , \tag{2.11}$$

where

$$\gamma_i(k) \triangleq (1-\alpha)\frac{\|\mathbf{w}(k)\|_1}{\mathrm{N}+1} + (1+\alpha)|w_i(k)| \tag{2.12}$$

and $\alpha \in [-1, 1)$. For $\alpha = -1$, the IPNLMS and NLMS algorithms are identical; and for $\alpha \approx 1$, the IPNLMS algorithm behaves like the PNLMS algorithm. In other words, α controls how proportionate will the IPNLMS algorithm be. As we have already explained, decreasing the degree of proportionality is interesting to obtain an algorithm that is robust to dispersive impulsive responses. We can interpret (2.12) as a sum of two terms: the first is an average of the adaptive coefficients, which balances the error introduced by the estimation (*NLMS* term); and the second is the *proportionate* term [39]. The operation of the IPNLMS algorithm is depicted in Algorithm 4.

Algorithm 4: The IPNLMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose μ in the range $0 < \mu \leq 1$

choose α in the range $-1 \leq \alpha < 1$

δ = small constant

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$

Do for $i \leq \mathrm{N}$

$\gamma_i(k) = (1-\alpha)\frac{\|\mathbf{w}(k)\|_1}{\mathrm{N}+1} + (1+\alpha)|w_i(k)|$

$g_i(k) = \frac{\gamma_i(k)}{\|\boldsymbol{\gamma}(k)\|_1}$

end for

Compute $\mathbf{G} = \mathsf{Diag}\{g_0\ \ldots\ g_{\mathrm{N}}\}$

$\mathbf{w}(k+1) = \mathbf{w}(k) + \frac{\mu\mathbf{G}(k)\mathbf{x}(k)e(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{G}(k)\mathbf{x}(k) + \delta}$

2.5 The ℓ_0-LMS Algorithm

In the last few years, a new family of algorithms that aims at exploit the sparsity of systems has appeared [51, 53]. It is the so-called regularized family. Algorithms of this type use a specific regularization function to promote sparsity on the adaptive filter coefficients. In this way, the sparsity is explicitly modeled.

The ℓ_0-LMS algorithm incorporates an approximation of the ℓ_0-norm penalty function in the cost function of the LMS algorithm [53, 54, 62]. Hence, with this technique, the system sparsity is clearly exploited by the ℓ_0-LMS algorithm. There-

fore, the new cost function is defined as

$$\xi(k) = e^2(k) + \gamma \|\mathbf{w}(k)\|_0, \tag{2.13}$$

where $\|\cdot\|_0$ represents the ℓ_0-norm, which is basically a function that counts the quantity of nonzero entries [51], and $\gamma > 0$ is a factor that weights the relevance of the new penalty function.

In practice, the implementation of the ℓ_0-norm function is not trivial. Due to the discontinuity of the ℓ_0-norm, it is common to use an approximation of the ℓ_0-norm by a continuous function. An usual approximation function is given by [51, 53]

$$\|\mathbf{w}(k)\|_0 \approx F_\beta[\mathbf{w}(k)] = \sum_{i=0}^{\mathrm{N}} \left(1 - \mathrm{e}^{-\beta|w_i(k)|}\right) \ , \tag{2.14}$$

where both sides of (2.14) are strictly equal when β tends to infinity [53]. Thus, (2.13) can be rewritten as

$$\xi(k) = e^2(k) + \gamma \sum_{i=0}^{\mathrm{N}} \left(1 - \mathrm{e}^{-\beta|w_i(k)|}\right) \ . \tag{2.15}$$

By solving (2.15), we have the update equation of the ℓ_0-LMS algorithm. Given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu \mathbf{x}(k) e(k) + \kappa \mathbf{f}_\beta\left[\mathbf{w}(k)\right] \ , \tag{2.16}$$

where $\kappa = \mu\gamma$ and $f_\beta\left[w_i(k)\right] \triangleq \dfrac{\partial F_\beta}{\partial w_i(k)}$ is written as follows

$$f_\beta\left[w_i(k)\right] = \beta \mathrm{sign}(w_i(k)) \mathrm{e}^{-\beta|w_i(k)|} \ , \tag{2.17}$$

where sign(.) is a component-wise sign function defined as

$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x > 0 \ , \\ 0, & \text{if } x = 0 \ , \\ -1, & \text{if } x < 0 \ . \end{cases} \tag{2.18}$$

We use the first order truncation of the Taylor series expansion of exponential functions to reduce the computational complexity of (2.17) [53, 63]. Then, $\mathbf{f}_\beta(.)$ is

simplified as

$$f_\beta(x) = \begin{cases} \beta^2 x + \beta, & \frac{-1}{\beta} \leq x < 0 \ ; \\ \beta^2 x - \beta, & 0 < x \leq \frac{1}{\beta} \ ; \\ 0, & \text{otherwise.} \end{cases} \tag{2.19}$$

The third term in the right side of (2.16) is the derivative of the Laplace function [51, 64], which can be interpreted as a *zero-attractor*. When a coefficient is in the range $\left[\frac{-1}{\beta}, \frac{1}{\beta}\right]$, this coefficient is attracted to zero by the ℓ_0-LMS algorithm. One can notice that this function is not convex, and the choice of β represents a trade-off between smoothness and quality of approximation [51]. A description of the operation of the ℓ_0-LMS algorithm is given in Algorithm 5.

Algorithm 5: The ℓ_0-LMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose a suitable μ

choose β

choose κ, 2×10^{-3} is a good value

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$

Compute $\mathbf{f}_\beta\left[\mathbf{w}(k)\right]$ as in (2.19)

$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu\mathbf{x}(k)e(k) + \kappa\mathbf{f}_\beta\left[\mathbf{w}(k)\right]$

2.6 The ℓ_0-NLMS Algorithm

Much like the LMS algorithm, the ℓ_0-LMS algorithm also has a normalized variation. For the same reasons and with analogous modifications, we can formulate the ℓ_0-NLMS algorithm [65]. The complete description of the ℓ_0-NLMS algorithm is depicted in Algorithm 6.

Algorithm 6: The ℓ_0-NLMS algorithm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose μ in the range $0 < \mu \leq 1$

choose κ, 2×10^{-3} is a good value

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{w}^{\mathrm{T}}(k)\mathbf{x}(k)$

Compute $\mathbf{f}_\beta\left[\mathbf{w}(k)\right]$ as in (2.19)

$\mathbf{w}(k+1) = \mathbf{w}(k) + \dfrac{\mu\mathbf{x}(k)e(k)}{\mathbf{x}^{\mathrm{T}}(k)\mathbf{x}(k) + \delta} + \kappa\mathbf{f}_\beta\left[\mathbf{w}(k)\right]$

Chapter 3

Properties of the Proportionate-type Algorithms

In this chapter, we address some properties of the proportionate-type algorithms. We design several experiments to base our conclusions. By discussing the results, one may notice that this family of algorithms cannot exploit the system sparsity, strictly speaking. For the sake of clarity, we inform the parameters for the simulations throughout the chapter, everytime a new experiment is described.

3.1 The Role of the Weighting Matrix

In Chapter 2, we presented some proportionate-type algorithms. Section 2.3 described the PNLMS algorithm and introduced the concept of the weighting matrix $\mathbf{G}(k)$, which is a diagonal matrix whose entries store some proportional relation with their corresponding adaptive filter coefficients [27]. Moreover, the matrix $\mathbf{G}(k)$ defines an algorithm of the proportionate-type family [27, 39–41]. Hence, (2.9) and (2.10) define the PNLMS algorithm, which is the most trivial manner to assign a proportional update. However, the IPNLMS algorithm adopts a different approach, where it introduces an *NLMS* term in its weighting matrix, given by (2.12). This new term addresses the problem of the poor performance of the PNLMS algorithm in dispersive systems [38–40].

We must consider some aspects before further analysis of the weighting matrix $\mathbf{G}(k)$. The initialization vector is not usually related to the system coefficients, then the behavior of $\mathbf{G}(k)$ can be unpredictable in the initialization process of the algorithms. Thus, we assume that the algorithm is close to the steady-state MSE. In other words, it has passed some iterations and the proportionate algorithm should be close to the optimum coefficients of the system. Besides that, we choose ρ and δ_p aiming that the entries in the diagonal of $\mathbf{G}(k)$ are proportionate to its respective

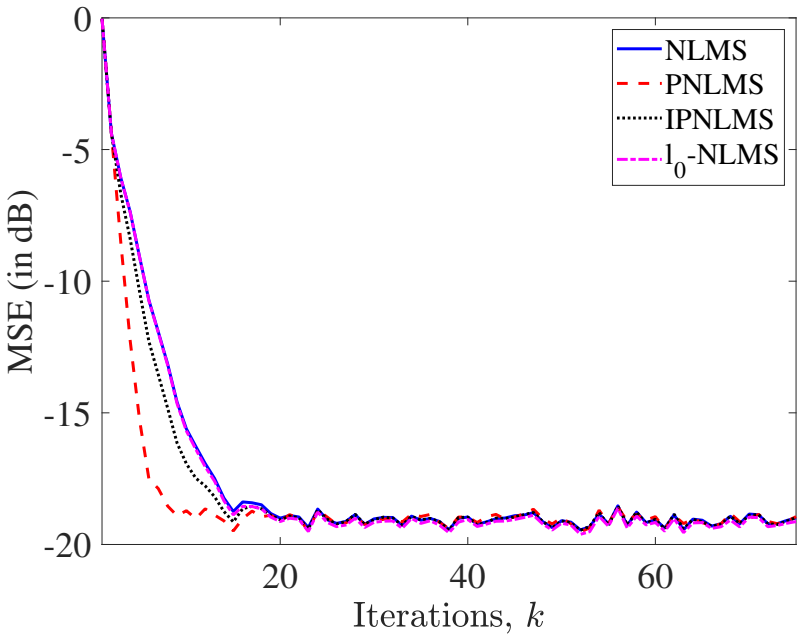
coefficients.

For simplicity, we use the weighting matrix of the PNLMS algorithm as the example for our analysis. By assuming an iteration close to the steady-state MSE, one can notice from (2.10) that $\gamma_i = |w_i(k)|$. Hence, (2.9) has the estimate of the coefficient in its dividend, and the ℓ_1-norm of the weight vector in the divisor, which stabilizes the algorithm [27]. In other words, coefficients with large magnitude receive larger values for $g_i(k)$, resulting in larger step sizes in comparison to the coefficients with small magnitude. Moreover, the high-magnitude coefficients can converge faster than the others. However, there is a trade-off between the convergence speed of high and low-magnitude coefficients. The matrix $\mathbf{G}(k)$ also makes small coefficients to converge slower, as it assigns smaller step sizes to these coefficients. Summarizing, as the relevant coefficients converge quickly to their optimum values, the low-magnitude coefficients are slower, taking many iterations to reach their optimum values.

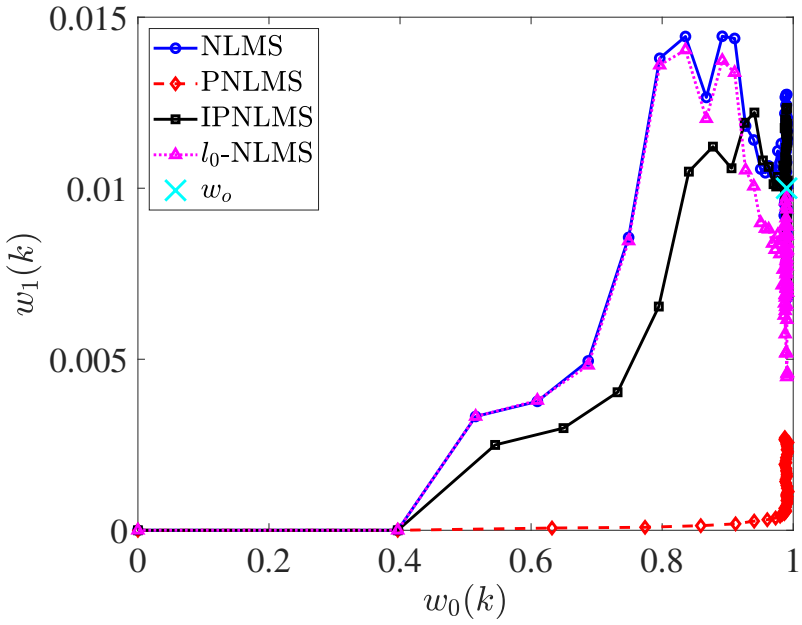
These are some of the reasons why a proportionate-type algorithm can achieve better performance than the NLMS algorithm in sparse scenarios. A sparse impulse response has a few non-zero coefficients, i.e., the total system energy concentrates in a small piece of the impulse response. Hence, the matrix $\mathbf{G}(k)$ gives large equivalent step sizes to these coefficients, making them converge very fast. Furthermore, we usually initialize an adaptive filtering algorithm with the null vector $\mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$, which favors the low-magnitude coefficients in sparse scenarios. The slow convergence speed of these coefficients is irrelevant when the initialization vector is near to their optimum values. Therefore, a proportionate-type algorithm does not take advantage of the number of zeroes present in a sparse impulse response. A proportionate algorithm exploits the large difference between the high-and low-magnitude coefficients. This feature is not unique to sparse systems, as any impulse responses can show large differences in the magnitude of its coefficients.

We propose some simple examples to show the influence of the matrix $\mathbf{G}(k)$ on the learning process of the PNLMS algorithm. We apply the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms to identify several unknown impulse responses. The ℓ_0-NLMS algorithm presents remarkable performance identifying sparse impulse responses [51, 53], which is one of the subjects of this work. Moreover, it belongs to the regularization family algorithms, then we can compare the performance of these two families of algorithms.

For the simulations in this section, the order of all unknown systems is 1, i.e., they have 2 coefficients. The input signal is a zero-mean binary phase-shift keying



(a) MSE learning curves.



(b) Convergence path.

Figure 3.1: MSE learning curves and convergence path of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_o^{(1)}$.

(BPSK), which has unit variance. [1] For the IPNLMS algorithm, we use $\alpha = -0.5$. For the ℓ_0-NLMS algorithm, we use the Laplacian form with first order truncation, we set the weight given to the approximation of the ℓ_0-norm as $\kappa = 2\times10^{-3}$, and the parameter that controls the quality of the ℓ_0-norm approximation is $\beta = 5$ [51, 53]. In all simulations, we set the step sizes $\mu = 0.4$ for all algorithms. The MSE

[1] For the sake of clarity in the results, we choose a BPSK signal as the system input. One of the simulations objectives is to evaluate the learning path of the adaptive coefficients for each algorithm. Thus, a BPSK input signal yields well-behaved learning paths, facilitating the visualization.

learning curves of the algorithms are computed by averaging the outcomes of 1000 independent trials.

3.1.1 First Experiment: Basic scenario

Suppose we want to identify the unknown impulse response $\mathbf{w}_{\mathrm{o}}^{(1)} = [0.99\ 0.01]^{\mathrm{T}}$. One can notice that $\mathbf{w}_{\mathrm{o}}^{(1)}$ is very simple, and it has only 2 coefficients, in which the first coefficient is much larger than the second one. Thus, we expect the PNLMS algorithm to converge faster to the optimum value of the larger coefficient. As for the smaller coefficient, the PNLMS algorithm updates with small step sizes until it reaches the vicinity of the low-magnitude coefficient.

Figure 3.1 depicts the MSE learning curves and convergence path of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(1)}$. In Figure 3.1a, one can notice the MSE learning curves of the algorithms, in which we set the step sizes of each algorithm to guarantee that they converge to the same steady-state MSE level. As we expect, the PNLMS algorithm shows the best performance, reaching the steady-state MSE faster than the other algorithms. Moreover, the IPNLMS algorithm takes a few more iterations than the PNLMS algorithm. Furthermore, the NLMS and ℓ_0-NLMS algorithms present the same and worst performance. This is an expected behavior of the ℓ_0-NLMS algorithm, i.e., since the unknown system is not sparse, the ℓ_0-NLMS algorithm behaves exactly as the NLMS algorithm.

Figure 3.1b presents the convergence path that the coefficients of the algorithms take to reach the coefficients of the optimum system. As we expect, the PNLMS algorithm quickly converges to the largest coefficient $w_{\mathrm{o}\,0}^{(1)} = 0.99$, next it goes in the $w_{\mathrm{o}\,1}^{(1)} = 0.01$ direction. One can notice that the PNLMS algorithm does not achieve a good precision in the smaller coefficient. [2] However, it achieves the best performance among the other algorithms, concluding that the largest coefficient controls the majority of the MSE in this system. The NLMS, IPNLMS and ℓ_0-NLMS algorithms do some combination between $w_0^{(1)}(k)$ and $w_1^{(1)}(k)$ to reach $\mathbf{w}_{\mathrm{o}}$, as expected [1, 3].

The role of the weighting matrix $\mathbf{G}(k)$ is changing the axes "size" of the system coefficients, shrinking the axis of the large coefficient, while it enlarges the axis of the small one. Figure 3.2 depicts the contours of the MSE surface for the impulse response $\mathbf{w}_{\mathrm{o}}^{(1)}$ without the influence of the matrix $\mathbf{G}(k)$, which the NLMS and ℓ_0-NLMS algorithms try to minimize. It is worth mentioning that the axes of Figure 3.2 are $\Delta w_i = w_i - w_{\mathrm{o}\,i}$. Thus, the minimum value of the MSE is at point the $(0, 0)$ as in [1].

[2] The PNLMS algorithm takes several iterations and, even so, it is far from the optimum value of the low-magnitude coefficient.

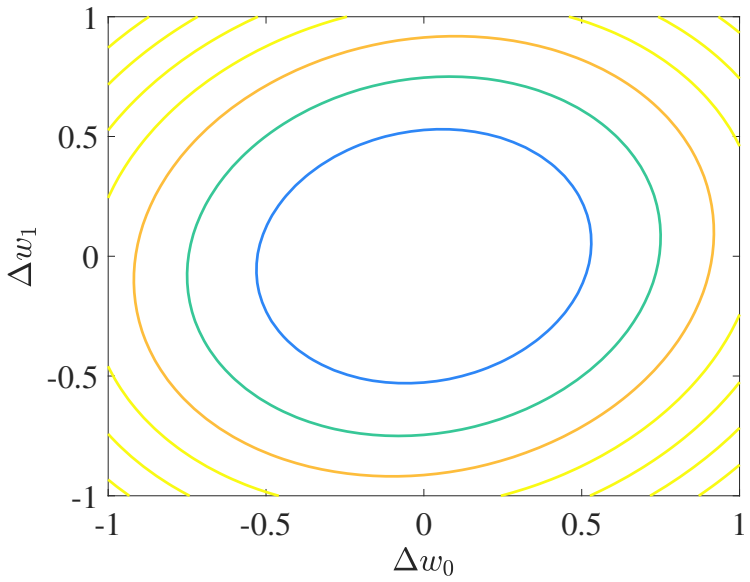
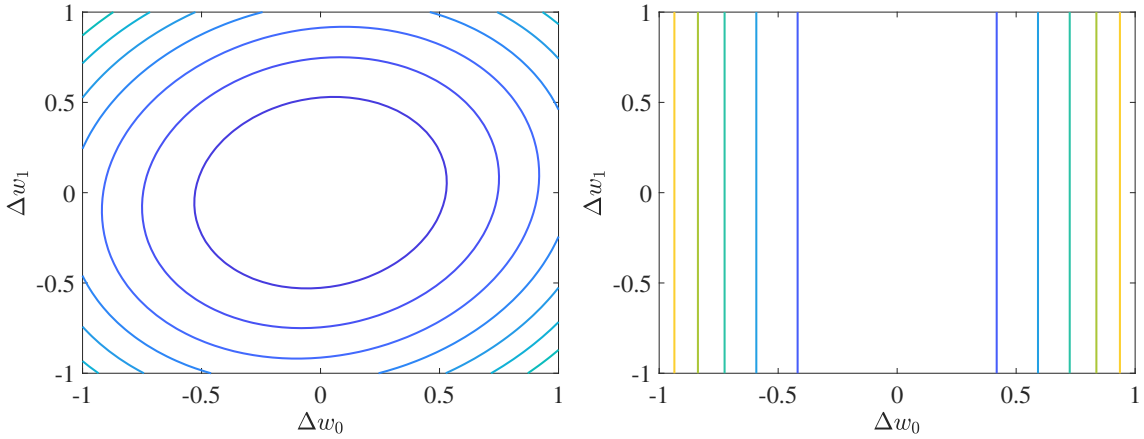


Figure 3.2: Contours of the MSE surface for the NLMS and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\text{o}}^{(1)}$.

The error surface contours presented in Figure 3.2 are intersections, in the MSE surface, by planes parallel to the $\mathbf{w}$ plane, placed at superior levels to the minimum error. The MSE can be expressed as a function of $\Delta\mathbf{w}$ as follows

$$\xi = \xi_{\text{min}} + \Delta\mathbf{w}^{\text{T}}\mathbf{R}\Delta\mathbf{w} \ , \tag{3.1}$$

where ξ_{min} is the minimum error and $\mathbf{R} = \mathbb{E}\big[\mathbf{x}(k)\mathbf{x}(k)^{\text{T}}\big]$ is the autocorrelation matrix of the input signal [1]. By setting fixed values to ξ in (3.1), we have the contours of the MSE surface.

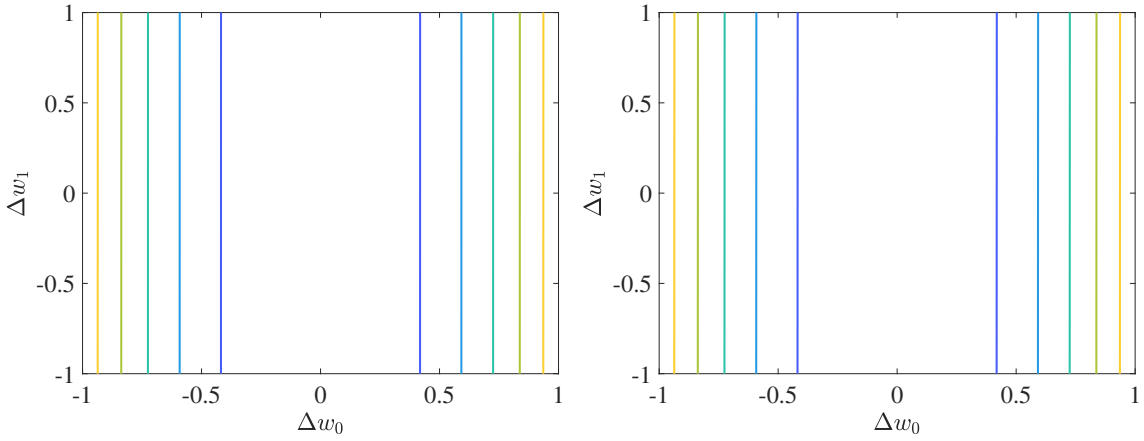


(a) $k = 1$. (b) $k = 2$.

Figure 3.3: Influence of the matrix $\mathbf{G}(k)$ on the contours of the MSE surface for the PNLMS algorithm considering $\mathbf{w}_{\text{o}}^{(1)}$, at the beginning of the learning process.

Figure 3.3 presents the contours of the MSE surface that the PNLMS algorithm tries to minimize, at the beginning of the learning process. Figure 3.3a shows the

contours of the MSE surface for $k = 1$. At this moment, the PNLMS algorithm does not have any knowledge about $\mathbf{w}_{\mathrm{o}}^{(1)}$. Then, the contours are almost the same as in Figure 3.2. The only difference between the figures is the error levels. In Figure 3.3a, the level curves are presenting a blue pattern, instead of a multicolor pattern. Thus, the MSE surface is smoother than the surface minimized by the NLMS and ℓ_0-NLMS algorithms. Figure 3.3b depicts these same contours, however for $k = 2$. One can notice that the contours completely change when the most relevant coefficient feeds the matrix $\mathbf{G}(k)$. The PNLMS algorithm shrinks the axis Δw_0, aiming to increase the convergence rate in that direction. Hence, the minimum point becomes a line, which is parallel to axis Δw_1 and across $\Delta w_0 = 0$. This explains why the PNLMS algorithm was not able to converge to the second coefficient of the unknown system $\mathbf{w}_{\mathrm{o}}^{(1)}$.



(a) $k = 30$. (b) $k = 75$.

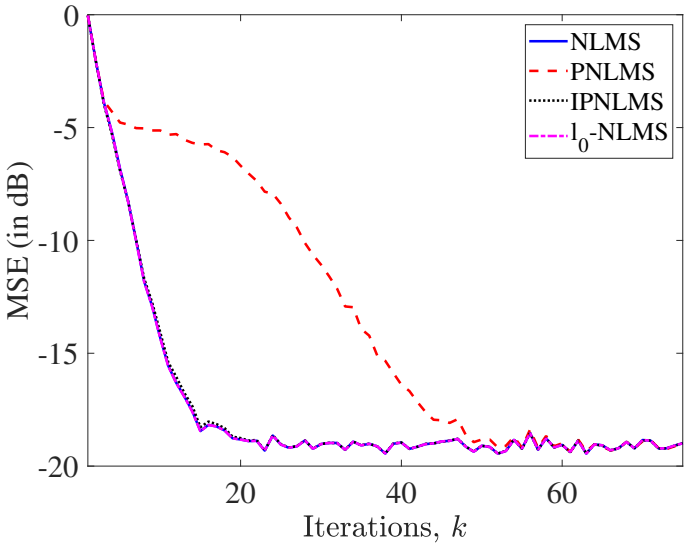
Figure 3.4: Influence of the matrix $\mathbf{G}(k)$ on the contours of the MSE surface for the PNLMS algorithm considering $\mathbf{w}_{\mathrm{o}}^{(1)}$ in the steady-state MSE.

Figure 3.4 shows the contours of the MSE surface of the PNLMS algorithm in the steady-state. In Figure 3.4a, we evaluate contours for $k = 30$ and we can see the same contours as in Figure 3.3b, which represents a parabolic cylinder [66]. Figure 3.4b depicts contours of the MSE surface for $k = 75$, and the result is the same as before. Hence, after converging to $w_{\mathrm{o}\,0}^{(1)}$, the PNLMS algorithm does not change its MSE surface, yielding the slow convergence speed to $w_{\mathrm{o}\,1}^{(1)}$. The matrix $\mathbf{G}(k)$ enlarges the axis Δw_1 impairing the convergence speed in that direction.

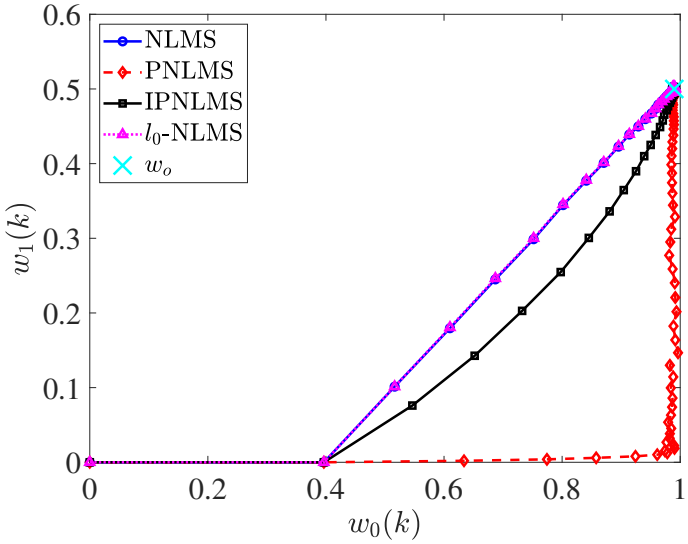
3.1.2 Second Experiment: Increasing $w_{\mathrm{o}\,1}^{(1)}$

For the second example, we consider the unknown impulse response $\mathbf{w}_{\mathrm{o}}^{(2)} = [0.99\ 0.5]^{\mathrm{T}}$. We decrease the difference between the coefficients by keeping the large coefficient and increasing the small one. Thus, we can verify the influence of $\mathbf{G}(k)$ when the difference of the coefficients is not so high. Figure 3.5a illustrates the MSE

learning curves of the algorithms for this system. One can notice that the PNLMS algorithm has the worst performance with the decrease of the difference between the coefficients. It takes about 60 iterations to reach the steady-state MSE, whereas the other algorithms, which have similar performance, take 20 iterations.



(a) MSE learning curves.



(b) Convergence path.

Figure 3.5: MSE learning curves and convergence path of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(2)}$.

Figure 3.5b shows the convergence path that the algorithms take to reach $\mathbf{w}_{\mathrm{o}}^{(2)}$. Unlike the result in Figure 3.1b, the PNLMS algorithm reaches values closer to the small coefficient $w_{\mathrm{o}\,1}^{(2)} = 0.5$, albeit, there is a trade-off. The PNLMS algorithm can achieve more precision converging to $w_{\mathrm{o}\,1}^{(2)}$ by sacrificing convergence speed in the direction of the $w_{\mathrm{o}\,0}^{(2)}$. As we expect, the PNLMS algorithm takes more iterations to reach $w_{\mathrm{o}\,0}^{(2)} = 0.99$. As the difference between the coefficients become small, the equivalent step sizes that $\mathbf{G}(k)$ assigns to the coefficients are smaller than those in the

first example. It is worth mentioning that the *NLMS* term present in the IPNLMS algorithm maintains its convergence path more similar to the NLMS algorithm.

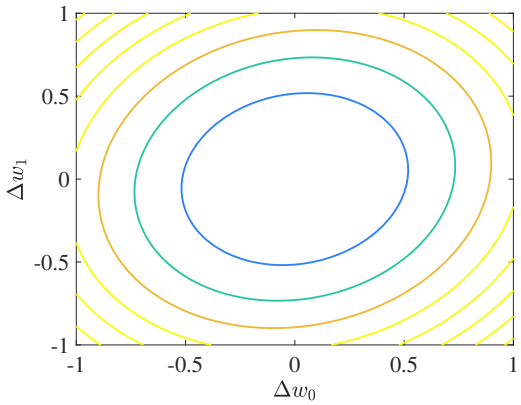
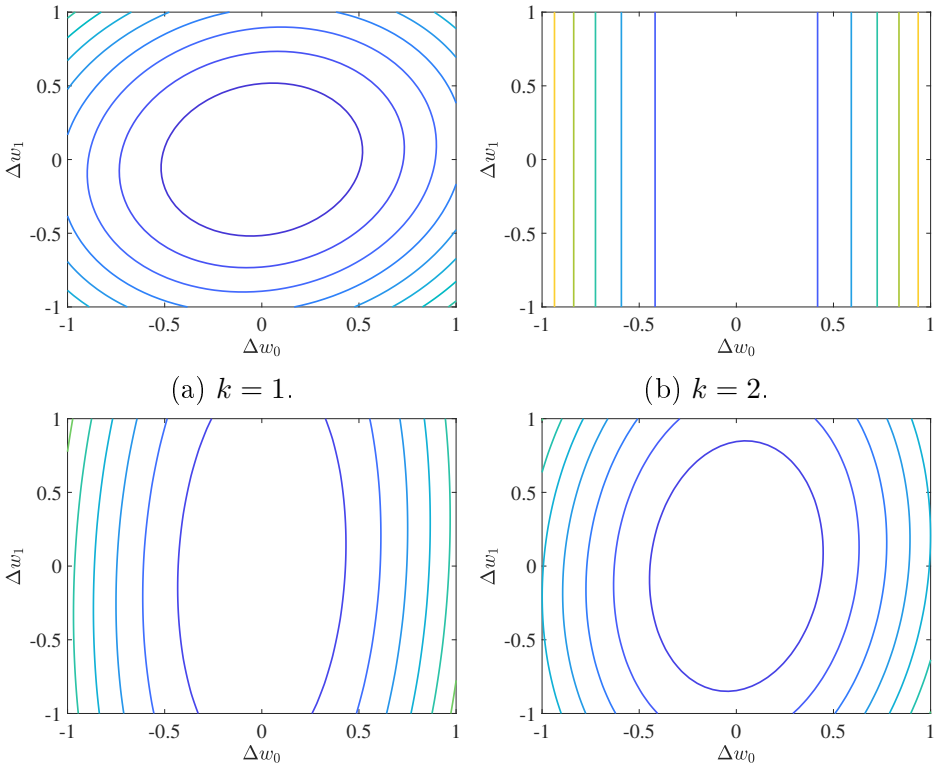


Figure 3.6: Contours of the MSE surface for the NLMS and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(2)}$.

Figure 3.6 presents the contours of the MSE surface for the unknown impulse response $\mathbf{w}_{\mathrm{o}}^{(2)}$, which both the NLMS and ℓ_0-NLMS algorithms minimize. The contours of the MSE surface in Figure 3.6 are very similar to the contours in Figure 3.2. As the weighting matrix $\mathbf{G}(k)$ does not influence the curves in Figure 3.6, we use these level curves as the reference for this simulation.



(a) $k = 1$. (b) $k = 2$.

(c) $k = 30$. (d) $k = 50$.

Figure 3.7: Influence of the matrix $\mathbf{G}(k)$ on the contours of the MSE surface for the PNLMS algorithm considering $\mathbf{w}_{\mathrm{o}}^{(2)}$, before the steady-state MSE.

Figure 3.7 depicts the contours MSE surface at different iterations before the PNLMS algorithm reaches the steady-state. In Figure 3.7a, we observe the level curves at iteration $k = 1$. The PNLMS algorithm is in its initial state and does not acknowledge the magnitude of the coefficients. Then, Figure 3.7a only shows a smoother version of the reference curves in Figure 3.6. Figure 3.7b shows contours for $k = 2$. One can notice that as soon as the PNLMS algorithm acknowledges the first coefficient, it shrinks the axis Δw_0, making the algorithm converges fast to $w_{\mathrm{o}\,0}^{(2)}$, as presented in Figure 3.5b. The PNLMS algorithm starts changing its level curves by shrinking the axis Δw_1, as one can notice in Figure 3.7c. Hence, after the PNLMS algorithm converges to the vicinity of $w_{\mathrm{o}\,0}^{(2)}$, it tries to increase its convergence speed in the direction of the $w_{\mathrm{o}\,2}^{(2)}$. Figure 3.7d shows contours of the MSE surface for $k = 50$, where the PNLMS algorithm is reaching the steady-state MSE. We can notice an ellipse, with its semi-minor axis almost parallel to Δw_0, while its semi-major axis is almost parallel to Δw_1.

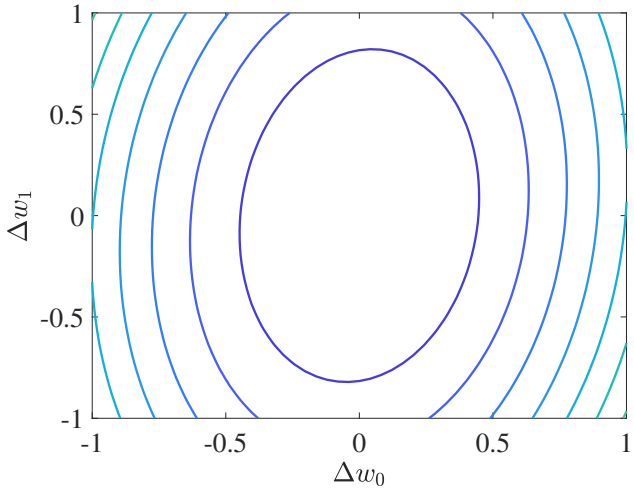


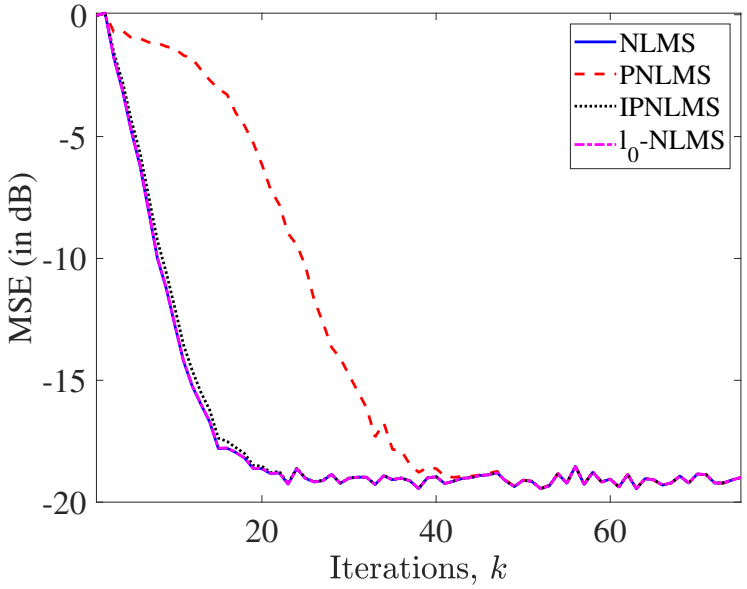
Figure 3.8: Influence of the matrix $\mathbf{G}(k)$ on the contours of the MSE surface for the PNLMS algorithm considering $\mathbf{w}_{\mathrm{o}}^{(2)}$ in the steady-state MSE.

One can notice in Figure 3.8 for $k = 75$ that the contours are the same as in Figure 3.7d. Then, these contours form the resultant MSE surface that the PNLMS algorithm minimizes, which are nothing like the Figure 3.6 that the NLMS and ℓ_0-NLMS algorithms try to minimize. The PNLMS algorithm changes the orientation of the axes aiming at a fast convergence in certain directions.

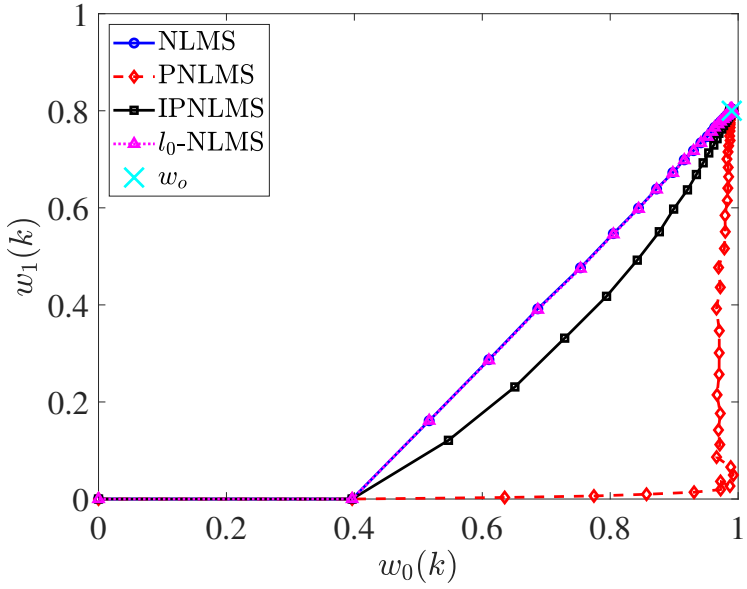
3.1.3 Third Experiment: Coefficients much closer

Our next example consists of identifying the unknown impulse response $\mathbf{w}_{\mathrm{o}}^{(3)} = [0.99\ 0.8]^{\mathrm{T}}$, whose both coefficients are very close to each other. We evaluate the matrix $\mathbf{G}(k)$ of the PNLMS algorithm when the system coefficients are even closer to each other. In Figure 3.9a, we present the MSE learning curves of the NLMS,

PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(3)} = [0.99\ 0.8]^{\mathrm{T}}$. As the PNLMS algorithm does not have a high proportionate factor to properly operate, it achieves the worst performance. The PNLMS algorithm take several iterations to reach the steady-state MSE, similar to the second experiment. The other algorithms have the same performance, all of them take about 20 iterations to reach the steady-state MSE.



(a) MSE learning curves.



(b) Convergence path.

Figure 3.9: MSE learning curves and convergence path of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(3)}$.

Figure 3.9b depicts the convergence path taking for each algorithm to converge to the optimum value $\mathbf{w}_{\mathrm{o}}^{(3)}$. We can notice that the PNLMS algorithm presents its conventional convergence path, converging to the large coefficient, and then, going to

the next one. However, the PNLMS algorithm converges with larger step sizes in the direction of the second coefficient $w_{\mathrm{o}\,1}^{(3)}$ in comparison to the latter examples. Hence, this increases the number of iterations that the PNLMS algorithm takes to reach $w_{\mathrm{o}\,0}^{(3)}$. The difference between the coefficients is so small that the PNLMS algorithms almost update in each direction with the same step sizes, albeit, it identifies one coefficient at a time.

Figure 3.10 shows the contours for the impulse response $\mathbf{w}_{\mathrm{o}}^{(3)}$ without the presence of the weighting matrix $\mathbf{G}(k)$. These contours are very similar to that in the previous examples, in which all of them are ellipsoids. We use these level curves as a reference to the next results.

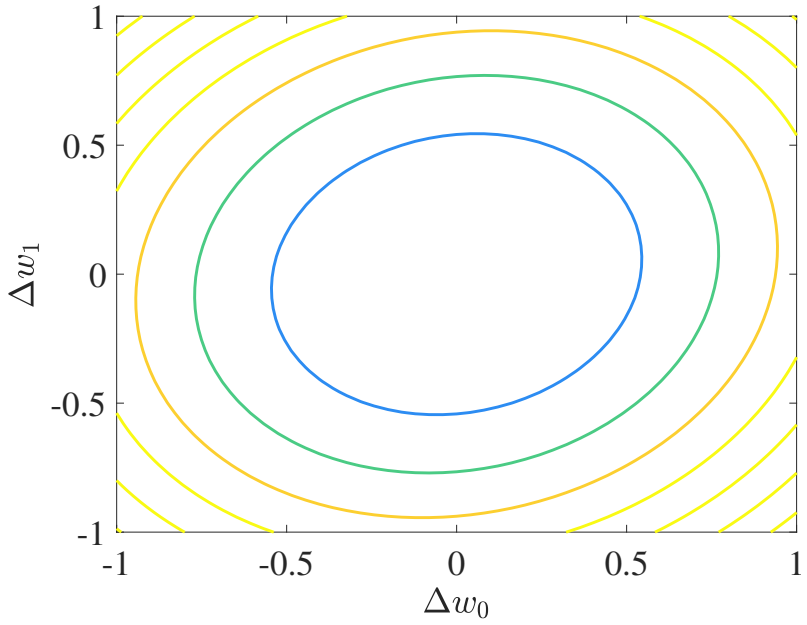
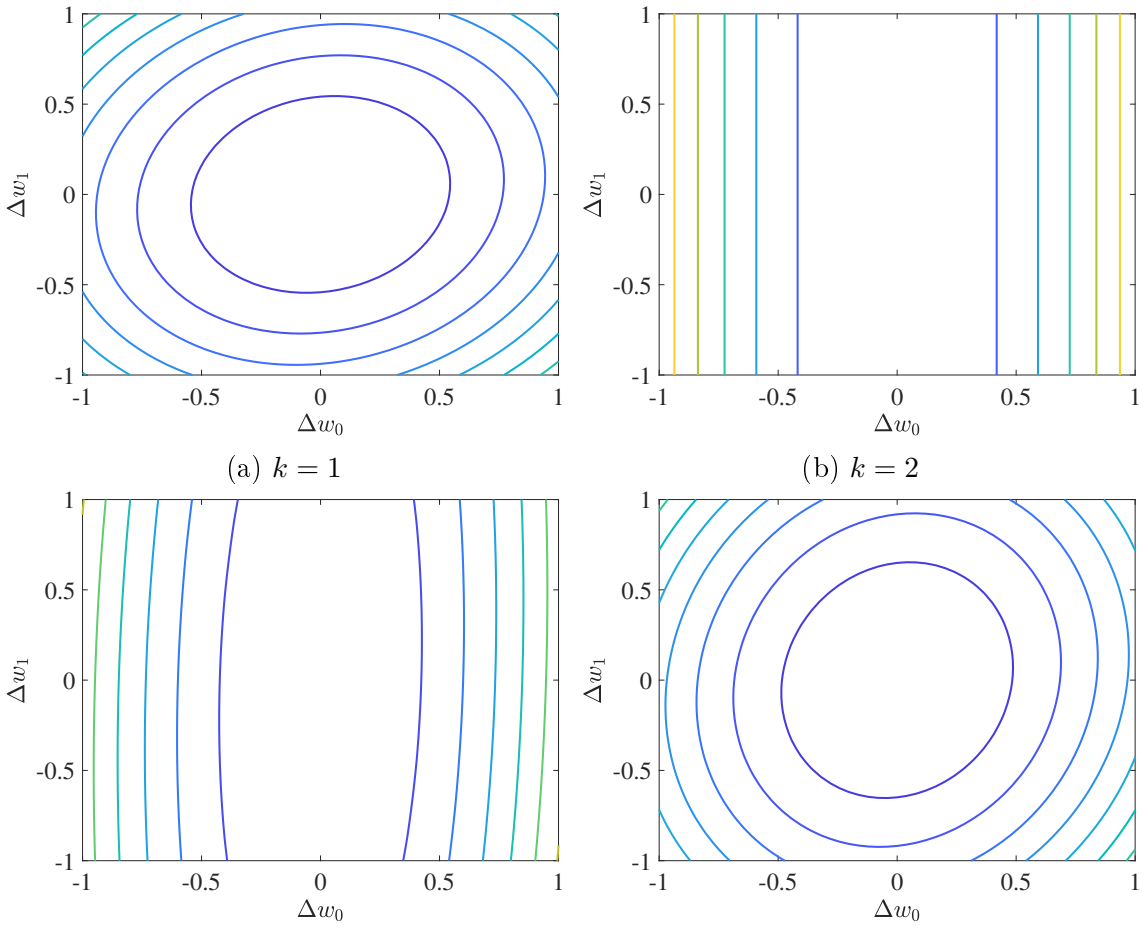


Figure 3.10: Contours of the MSE surface for the NLMS and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(3)}$.

Figure 3.11 presents several contours of the MSE surface when the PNLMS algorithm is identifying the impulse response. Figure 3.11a depicts the level curves at the very beginning of the learning process, at iteration $k = 1$. At this point of the process, the PNLMS algorithm does not sense the coefficients to distinguish their magnitude. Therefore, $\mathbf{G}(k)$ attenuates the adaptive coefficients making a smoother version of the MSE surface present in Figure 3.10. At the moment that the first coefficient feeds the PNLMS algorithm, $\mathbf{G}(k)$ shrinks Δw_0 and enlarges Δw_1, changing the MSE surface to a parabolic cylinder. This modification in the level curves favors the convergence speed in the direction of the $w_{\mathrm{o}\,0}^{(3)}$. Figure 3.11b presents the contours that forms this surface, which the PNLMS algorithm aims at quickly identify the coefficient $w_{\mathrm{o}\,0}^{(3)}$. However, it takes a few iterations to the PNLMS algorithm to change its MSE surface once more. One can notice that in Figure 3.11c, the level curves start to shrink in the axis Δw_1. At iteration $k = 15$,



(a) $k = 1$ (b) $k = 2$

(c) $k = 15$ (d) $k = 30$

Figure 3.11: Influence of the matrix $\mathbf{G}(k)$ on the contours of the MSE surface for the PNLMS algorithm considering $\mathbf{w}_{\mathrm{o}}^{(3)}$, before the steady-state MSE.

the PNLMS algorithm takes the direction to $w_{\mathrm{o}\,1}^{(3)}$. Figure 3.11d shows the contours for $k = 30$, in a few iteration before the PNLMS algorithms reaches the steady-state MSE. As we expect, the contours are similar to a circle, considering that both coefficients are close to each other.

Figure 3.12 depicts the contours of the MSE surface at iteration $k = 75$. We can notice that the MSE surface is very similar to that formed by the contours in Figure 3.11d. The PNLMS algorithm ends minimizing the MSE surface formed by the contours in Figure 3.12. By doing several changes in its error function, it tries different directions to achieve fast convergence speed for the adaptive coefficients.

3.1.4 Section Remark

The weighting matrix $\mathbf{G}(k)$ introduces an equivalent step size that is proportional to the absolute value of the coefficients. In other words, high-magnitude coefficients receive larger step sizes, converging faster than the low-magnitude ones, which receive smaller step sizes. Thus, the PNLMS algorithm exploits the large difference

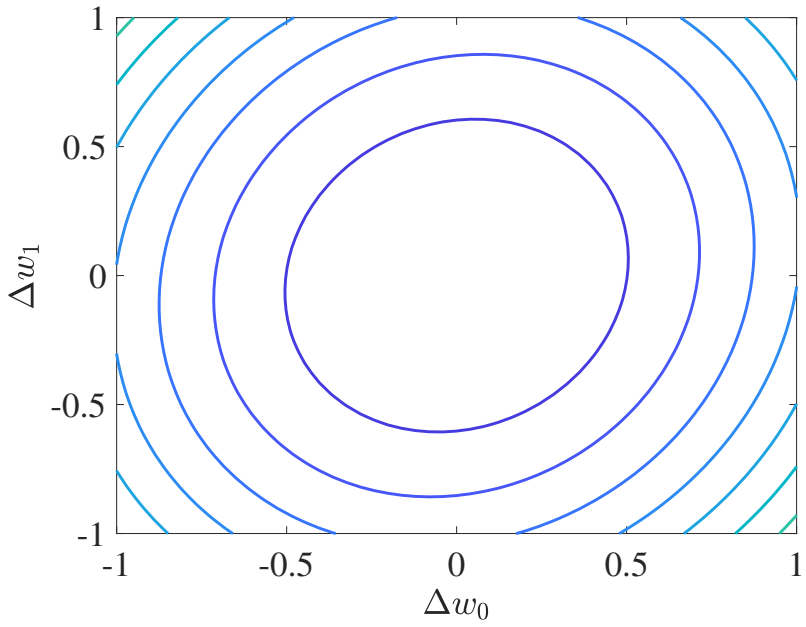


Figure 3.12: Influence of the matrix $\mathbf{G}(k)$ on the contours of the MSE surface for the PNLMS algorithm considering $\mathbf{w}_{\text{o}}^{(3)}$ in the steady-state MSE.

between the magnitude of the system coefficients. Therefore, it exploits something more general than the sparsity in the system.

3.2 Initialization is Critical

As an adaptive filtering algorithm minimizes a cost function, it needs a given point to start the learning process, the so-called initialization vector. A proper choice for the system initialization should be a vector close to the minimum value of the objective function. However, that is not possible without any prior knowledge about the cost function. Several research efforts prove that the convergence of an adaptive filter is independent of the vector that initializes the algorithm [1, 3]. Then, we use the vector $\mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\text{T}}$ as the algorithm initialization when we do not have any prior information about the problem. Indeed, in a system identification problem where the impulse response is sparse, i.e., most coefficients of the impulse response are zeroes, then $\mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\text{T}}$ is a suitable choice as the initialization vector. Thus, by starting the algorithm with $\mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\text{T}}$, it is already close to the objective function minimum.

A proportionate-type algorithm takes advantage of this usual initialization. One must remember that the proportionate idea relies on giving large step sizes to larger adaptive coefficients and small step sizes to the smaller ones [27, 39, 40, 62]. The algorithms assign these steps by the difference between the high-and low-magnitude coefficients. However, as they initialize with the vector $\mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\text{T}}$, the algorithms already start in the vicinity of most of the coefficients, making small

updates in each iteration. Moreover, as the dominant coefficients receive larger step sizes, the algorithms converge quickly to them. Hence, the PNLMS algorithm usually presents better performance than the NLMS algorithm for identifying sparse systems [27].

The slow convergence of the low-magnitude coefficients can impair the performance of the proportionate-type algorithms when these coefficients are not zeroes, or when we choose an initialization vector that is not close to them. In other words, the initialization vector must be close to the low-magnitude coefficients. When a proportionate-type algorithm converges to the most relevant coefficients, it assigns really small equivalent step sizes to the low-magnitude ones. In case we initialize with a bad vector, the proportionate-type algorithms converge slowly to these coefficients, which severely reduces their convergence speed. In cases like this, a proportionate-type algorithm, like the PNLMS algorithm, can perform worse than the NLMS algorithm.

We present several sparse impulse responses to validate this hypothesis. The simulations compare the performance of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms when we initialize them with the usual vector $\mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$ and some other initialization vector that is far from zero. For the simulations in this section, the order of all unknown systems is 63, i.e., they have 64 coefficients. The input signal is a zero-mean white Gaussian noise with unit variance. For the IPNLMS algorithm, we use $\alpha = -0.5$. As for the simulations in the previous section, we use the Laplacian form with first order truncation for the ℓ_0-NLMS, we set $\kappa = 2 \times 10^{-3}$, and $\beta = 5$ [51, 53]. In all simulations, we set the step sizes for the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms to 0.4, 0.3, 0.4, and 0.99. The MSE learning curves of the algorithms are computed by averaging the outcomes of 1000 independent trials.

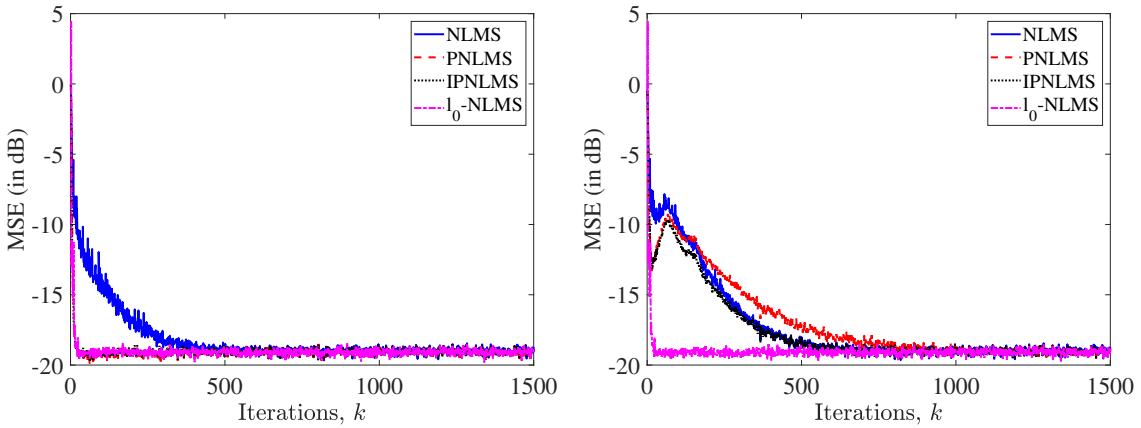
3.2.1 First Experiment: An ideal scenario

Suppose we want to identify the following sparse impulse response

$$\mathbf{w}_{\mathrm{o}}^{(4)}(n) = \begin{cases} 1, & \text{if } n = 0\ , \\ 0, & \text{if } 1 \leq n \leq 63\ . \end{cases} \tag{3.2}$$

The impulse response $\mathbf{w}_{\mathrm{o}}^{(4)}$ is an ideal sparse system; where the first coefficient is one, and the others are zeroes [67, 68]. The impulse response in (3.2) represents the perfect scenario for algorithms that exploit the sparsity of systems.

Figure 3.13 depicts the MSE learning curves of the NLMS, PNLMS, IPNLMS and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(4)}$. In Figure 3.13a, we initialize the algorithms



(a) $\mathbf{w}(0) = [0 \ \ldots \ 0]^{\mathrm{T}}$. (b) $\mathbf{w}(0) = [0.05 \ \ldots \ 0.05]^{\mathrm{T}}$.

Figure 3.13: MSE learning curves for different initialization vectors of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(4)}$.

with the vector $\mathbf{w}(0) = [0 \ \ldots \ 0]^{\mathrm{T}}$. As we expect, the proportionate-type and regularization algorithms reach the steady-state MSE quickly, just taking a few iterations to converge. One can notice that the remarkable fast initial convergence of the PNLMS and IPNLMS algorithms makes them achieve the best performance along the ℓ_0-NLMS algorithm. The NLMS algorithm shows the worst performance taking several iterations to reach the steady-state MSE.

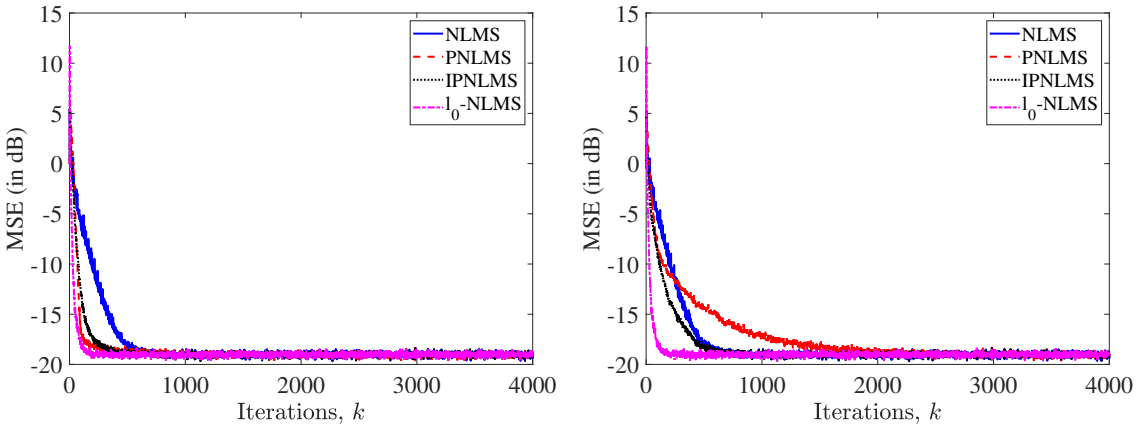
Figure 3.13b shows the MSE learning curves of the algorithms when we initialize them with $\mathbf{w}(0) = [0.05 \ \ldots \ 0.05]^{\mathrm{T}}$. The results that the PNLMS and IPNLMS algorithms reach are worse when comparing to the learning curves present in Figure 3.13a. In Figure 3.13b, one can notice that the IPNLMS algorithm matches its performance with the NLMS algorithm. Moreover, the PNLMS algorithm reaches the steady-state MSE slower than the NLMS algorithm. The initialization with the vector $\mathbf{w}(0) = [0.05 \ \ldots \ 0.05]^{\mathrm{T}}$ does not seem to affect the ℓ_0-NLMS algorithm, at least nothing that one can notice by analyzing Figure 3.13.

3.2.2 Second Experiment: Increasing the number of relevant coefficients

The impulse response given by (3.2) is very simple. It concentrates the entire energy in a single coefficient, which does not represent common real systems. Usually, a practical system has an impulse response with more than 1 non-zero coefficient. Hence, for the next simulation, we consider the following impulse response

$$\mathbf{w}_{\mathrm{o}}^{(5)}(n) = \begin{cases} 1, & \text{if } 0 \leq n \leq 3 \ , \\ 0, & \text{if } 4 \leq n \leq 63 \ . \end{cases} \tag{3.3}$$

Indeed, the impulse response $\mathbf{w}_{\text{o}}^{(5)}$ has only 4 non-zeroes coefficients, not much different from the $\mathbf{w}_{\text{o}}^{(4)}$. However, we want to keep a high sparsity degree for the proportionate algorithms. Moreover, if we have a total of 64 coefficients, 4 large coefficients are more than 5% of the total. Figure 3.14 presents the MSE learning curves of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering the impulse response $\mathbf{w}_{\text{o}}^{(5)}$.



(a) $\mathbf{w}(0) = [0 \ \ldots \ 0]^{\text{T}}$. (b) $\mathbf{w}(0) = [0.05 \ \ldots \ 0.05]^{\text{T}}$.

Figure 3.14: MSE learning curves for different initialization vectors of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\text{o}}^{(5)}$.

In Figure 3.14a, one can notice that the regularization and proportionate-type algorithms reach the steady-state MSE in a few iterations. By initializing the algorithms with $\mathbf{w}(0) = [0 \ \ldots \ 0]^{\text{T}}$, the PNLMS algorithm achieves almost the same performance as the ℓ_0-NLMS algorithm, as we expect for the fast initial convergence of the PNLMS algorithm in a sparse system like $\mathbf{w}_{\text{o}}^{(5)}$. Figure 3.14a depicts the MSE learning curves of the algorithms when we initialize them with the vector $\mathbf{w}(0) = [0.05 \ \ldots \ 0.05]^{\text{T}}$. The new initialization vector degrades the convergence speed of the PNLMS and IPNLMS algorithms. The PNLMS algorithm takes several iterations to converge to the steady-state MSE, performing worse than the NLMS algorithm. The IPNLMS algorithm loses its fast initial convergence, which makes its curve shape more similar to the MSE learning curve of the NLMS algorithm. Despite the new initialization vector, the regularization algorithm keeps its performance, i.e., the "bad" initialization does not affect the ℓ_0-NLMS algorithm.

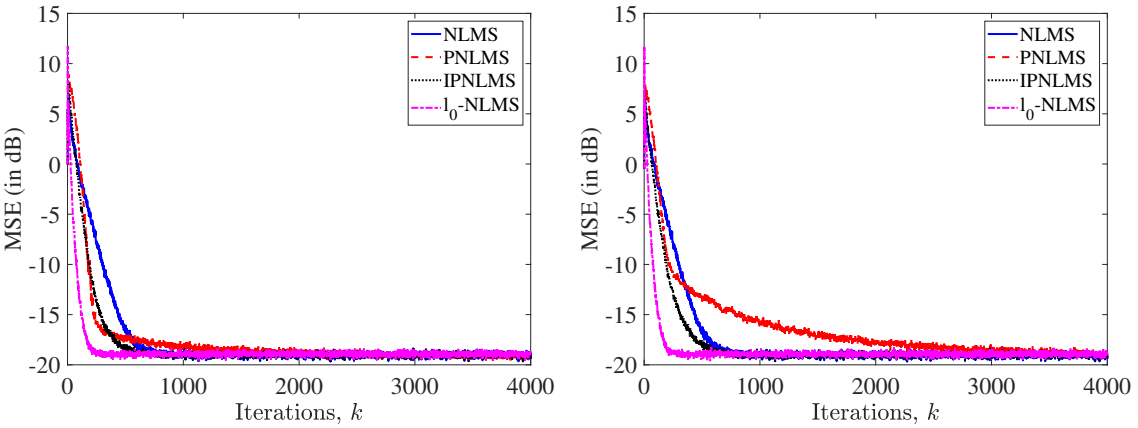
3.2.3 Third Experiment: Decreasing even more the sparsity degree

We want to verify the influence of a bad initialization when the system has an impulse response with even more relevant coefficients. As we increase the number of relevant coefficients, the system becomes less sparse. Hence, we need to identify

the following unknown impulse response

$$\mathbf{w}_{\mathrm{o}}^{(6)}(n) = \begin{cases} 1, & \text{if } 0 \leq n \leq 7 \ , \\ 0, & \text{if } 8 \leq n \leq 63 \ . \end{cases} \tag{3.4}$$

The impulse response $\mathbf{w}_{\mathrm{o}}^{(6)}$ has two times more relevant coefficients in comparison to $\mathbf{w}_{\mathrm{o}}^{(5)}$. In this experiment, 12.5% of the coefficients are relevant, i.e., the impulse response $\mathbf{w}_{\mathrm{o}}^{(6)}$ continues to have a considerable sparsity level. For example, impulse responses with this number of relevant coefficients represent echo-path channels [27, 41, 60]. Figure 3.15 shows the MSE learning curves of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering the unknown impulse response $\mathbf{w}_{\mathrm{o}}^{(6)}$.



(a) $\mathbf{w}(0) = [0 \ \ldots \ 0]^{\mathrm{T}}$. (b) $\mathbf{w}(0) = [0.05 \ \ldots \ 0.05]^{\mathrm{T}}$.

Figure 3.15: MSE learning curves for different initialization vectors of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(6)}$.

Figure 3.15a depicts the MSE learning curves of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms by initializing with the vector $\mathbf{w}(0) = [0 \ \ldots \ 0]^{\mathrm{T}}$. As we expect, all algorithms take a few iterations to converge to the steady-state MSE. However, the ℓ_0-NLMS algorithm converges much faster than the other algorithms. The performance of the IPNLMS, NLMS and PNLMS algorithms follows the ℓ_0-NLMS algorithm, respectively. The PNLMS algorithm shows its fast initial convergence speed, however, the convergence rate decreases, and it reaches the steady-state MSE slightly slower than the NLMS algorithm.

In Figure 3.15b, we can notice that by initializing the algorithms with $\mathbf{w}(0) = [0.05 \ \ldots \ 0.05]^{\mathrm{T}}$, the performance of the PNLMS and IPNLMS algorithms severely degrade. We observe the same results as for the previous examples. The PNLMS algorithm takes much more iterations to reach the steady-state MSE, performing worse than the NLMS algorithm. The shape of the MSE learning curve of the IPNLMS algorithm is closer to the shape of the NLMS curve, i.e., the bad initialization also impairs the performance of the IPNLMS algorithm. The NLMS and

ℓ_0-NLMS algorithms do not present any remarkable change by initializing with different vectors.

3.2.4 Section Remark

A proportionate-type algorithm requires an initialization vector that is close to the optimum value of the low-magnitude coefficients. A bad initialization vector makes the algorithm to converge slowly to the steady-state.

3.3 The Time-shifting Problem

The shift of an impulse response can be a problem for the PNLMS algorithm, due to the slow convergence rate of the low-magnitude coefficients. We will briefly review some steps of the learning process of an adaptive filter to understand how a shifted system impairs the performance of the PNLMS algorithm. Typically, an adaptive filter is represented by the block diagram presented in Figure 3.16, for system identification applications.

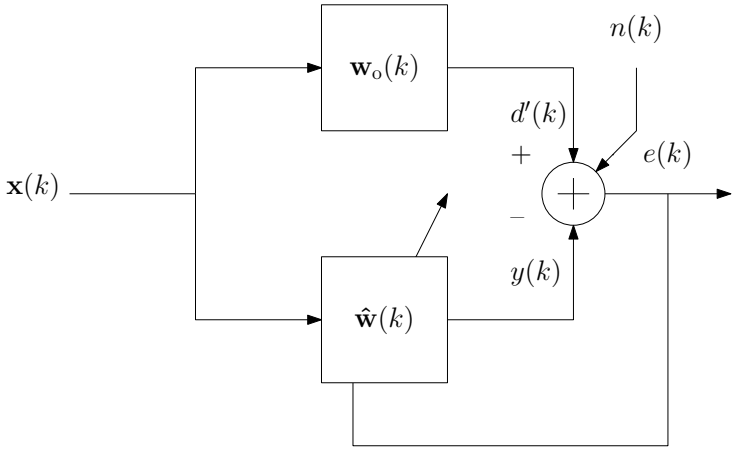


Figure 3.16: System identification adaptive filter.

Figure 3.16 depicts a classical way to describe the operation of an adaptive filter in a system identification problem. One can notice that the input signal $\mathbf{x}(k)$ is feeding the unknown impulse response $\mathbf{w}_o(k)$ and the adaptive filter $\hat{\mathbf{w}}(k)$; the noiseless desired signal $d'(k)$, the measurement noise $n(k)$ and the filter output $y(k)$ are producing the error signal $e(k)$, which updates the filter parameters. Figure 3.17 shows a closer view from the upper side of Figure 3.16. Figure 3.17 presents the input signal and the coefficients of the unknown impulse response. One can notice a serial-to-parallel process, in which the serial input signal passes over a tapped delay line to produce a parallel input that feeds the unknown system. This process

requires some initial conditions to operate in the first iterations that we solve by simply treating the input with negative indexes as zeroes, i.e, $x(i) = 0,\ \forall\ i < 0$.

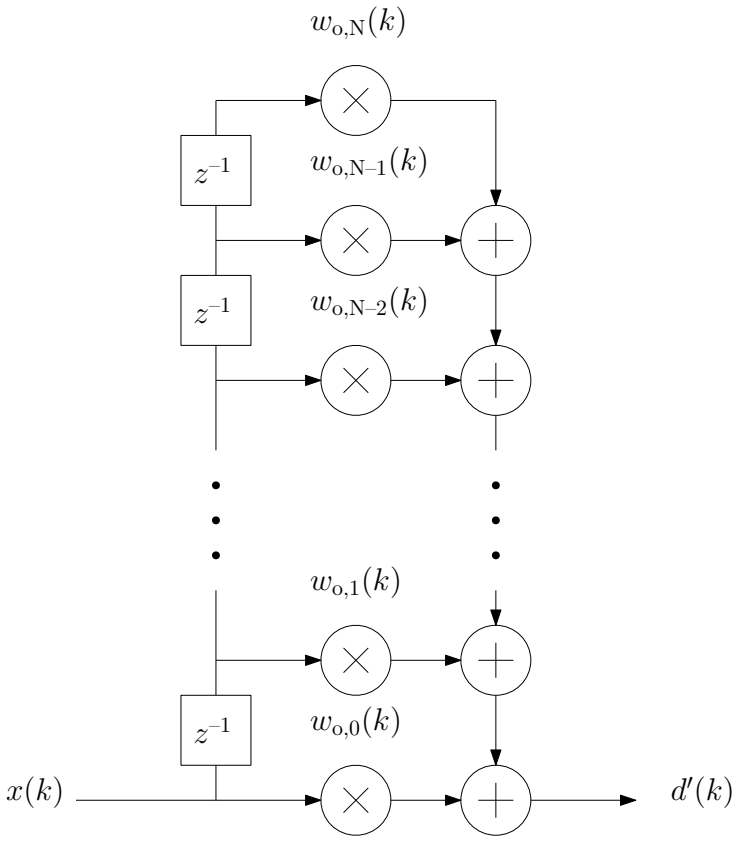


Figure 3.17: Tapped delay line applied to an impulse response.

Suppose we have the following sparse unknown system

$$\mathbf{w}_{o}^{(7)}(n) = \begin{cases} 0, & \text{if } 0 \leq n \leq 59\ , \\ 1, & \text{if } 60 \leq n \leq 63\ . \end{cases} \tag{3.5}$$

The impulse response in (3.5) is very similar to the system in (3.3). However, in $\mathbf{w}_{o}^{(7)}$, the last four coefficients are the most relevant. In fact, one can interpret the impulse response $\mathbf{w}_{o}^{(7)}$ as $\mathbf{w}_{o}^{(5)}$ with a shift of 59 time samples. From Figure 3.17, one can notice that by using a tapped delay line as the input of the system $\mathbf{w}_{o}^{(7)}$, $d'(k)$ is zero until the input signal sensitizes one of the non-zero coefficients of $\mathbf{w}_{o}^{(7)}$, which occurs at iteration $k = 60$, i.e, $d'(k)$ is different from zero when $x(0)$ multiplies $w_{o}^{(7)}(60)$. Hence, the adaptive filter can combine a relevant coefficient of $\mathbf{w}_{o}^{(7)}$, then starts to learn the unknown impulse response and stops to learn only noise, [3] i.e, until the iteration $k = 60$, the algorithm updates only considering the noise, consequently,

[3] In Figure 3.16, one can notice that the desired signal is $d(k) = d'(k) + n(k)$, if $d'(k) = 0$ then $d(k) = n(k)$, i.e., the desired signal is composed only by the measurement noise. Therefore, the adaptive filter output is only noise. One should remember that the adaptive filter tries to minimize $e(k) = d(k) - y(t)$.

updating the coefficients to values far from the optimum, disturbing the initialization vector.

This initialization period can be harmful to the PNLMS algorithm. This process changes the former initialization vector, generating another one. This new "initialization" vector is a combination of the measurement noise and the input signal. Hence, we do not have any knowledge about the system. Normally, this randomness on the vector promotes a bad initialization, which is critical for the PNLMS algorithm. Furthermore, when the adaptive filter updates the first relevant coefficient, locally, the other coefficients update to values far from the optimum, depending on the gradient of the function. In a sparse scenario, a few coefficients accumulate the majority of the system energy, as most of them are null. In our example, the last of them concentrates most of the energy. Then, as soon as the first relevant coefficient weighs the input, the error signal drastically increases, which compels the algorithm to abruptly alter the gradient path. After that, the coefficients can update to values far from the optimum, harming the proportionate behavior of the proportionate-type algorithms, mainly the PNLMS algorithm.

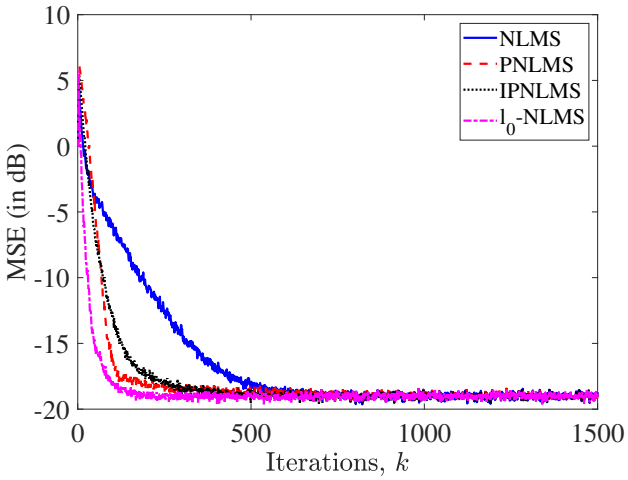
We design some simulations aiming for the comparison between a bad initialization of the algorithms and a shift in time of the impulse response. For the simulations in this section, the order of the systems is 63, i.e., they have 64 coefficients. All the parameters are the same as for the previous simulations. The MSE learning curves of the algorithms are computed by averaging the outcomes of 1000 independent trials.

3.3.1 First Experiment: Revisiting $\mathbf{w}_{\mathrm{o}}^{(5)}$

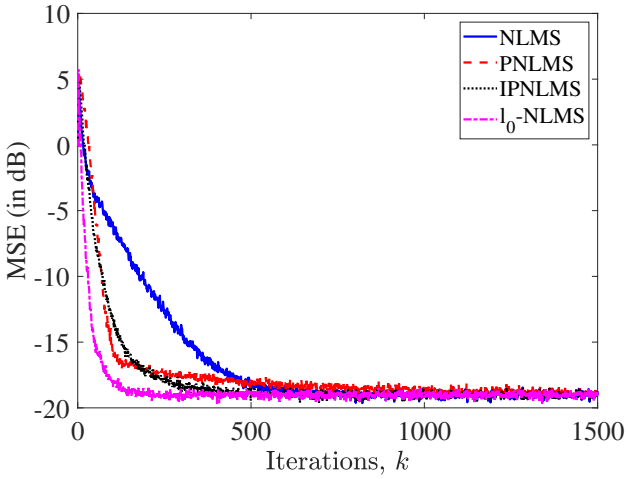
Figure 3.18 depicts the MSE learning curves of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering the impulse response (3.3) and its shifted version (3.5). One can notice that Figure 3.18a is the same as Figure 3.14a. As expected, when we initialize the algorithms with the vector $\mathbf{w}(0) = [0 \dots 0]^{\mathrm{T}}$, all algorithms converge quickly to the steady-state MSE, the exception is the NLMS algorithm. The ℓ_0-NLMS algorithm achieves the best performance, following by the PNLMS, IPNLMS and NLMS algorithms, respectively.

Figure 3.18b presents the MSE learning curves of the algorithms by using the initialization vector $\mathbf{w}(0) = [0.01 \dots 0.01]^{\mathrm{T}}$. This new initialization is sufficient to degrade the convergence speed of the PNLMS algorithm, as one can observe in Figure 3.18b. The PNLMS algorithm converges to the steady-state MSE even slower than the NLMS algorithm. It is worth mentioning that this minor misplacement is not sufficient to impair the performance of the IPNLMS algorithm. As expected, the NLMS and ℓ_0-NLMS algorithms are robust to this type of problem.

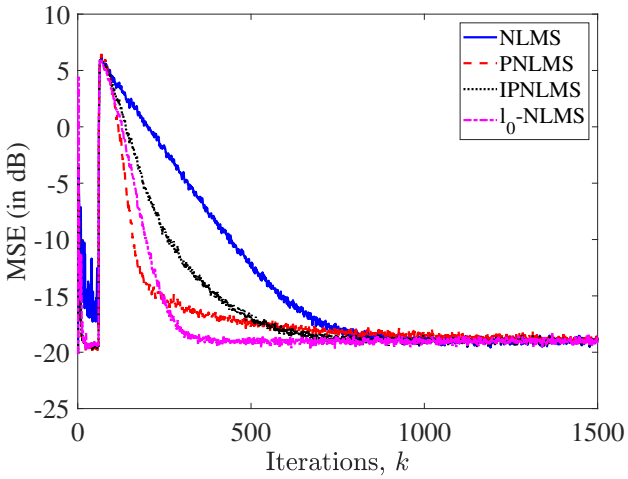
Figure 3.18c presents the MSE learning curves of the algorithms considering



(a) Considering $\mathbf{w}_{\mathrm{o}}^{(5)}$ with initialization vector $\mathbf{w}(0) = [0 \dots 0]^{\mathrm{T}}$.



(b) Considering $\mathbf{w}_{\mathrm{o}}^{(5)}$ with initialization vector $\mathbf{w}(0) = [0.01 \dots 0.01]^{\mathrm{T}}$.



(c) Considering $\mathbf{w}_{\mathrm{o}}^{(7)}$ with initialization vector $\mathbf{w}(0) = [0 \dots 0]^{\mathrm{T}}$.

Figure 3.18: MSE learning curves of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(5)}$ and its version shifted in time $\mathbf{w}_{\mathrm{o}}^{(7)}$.

the time-shifted impulse response (3.5). One can notice the similarity with the Figure 3.18b. Indeed, except for first iterations (the adaptive filters are filling the tapped delay line), the performance of the PNLMS algorithm is very similar to the case when we initialize with a bad vector. The PNLMS algorithm sustains its fast convergence speed in the "first" iterations [39, 40], however, in the end, it converges to steady-state MSE slower than the NLMS algorithm. The IPNLMS algorithm loses its fast initial convergence approximating the shape of its MSE curve to the NLMS algorithm, similarly to the results presented in Section 3.2. The NLMS and ℓ_0-NLMS algorithms do not show any remarkable changes.

3.3.2 Second Experiment: A more sparse scenario

In the next simulations, we use a sparser impulse response, i.e, the system energy concentrates on fewer coefficients. Suppose we want to identify the following impulse response

$$\mathbf{w}_{\mathrm{o}}^{(8)}(n) = \begin{cases} 1, & \text{if } 0 \leq n \leq 2 \ , \\ 0, & \text{if } 3 \leq n \leq 63 \ , \end{cases} \tag{3.6}$$

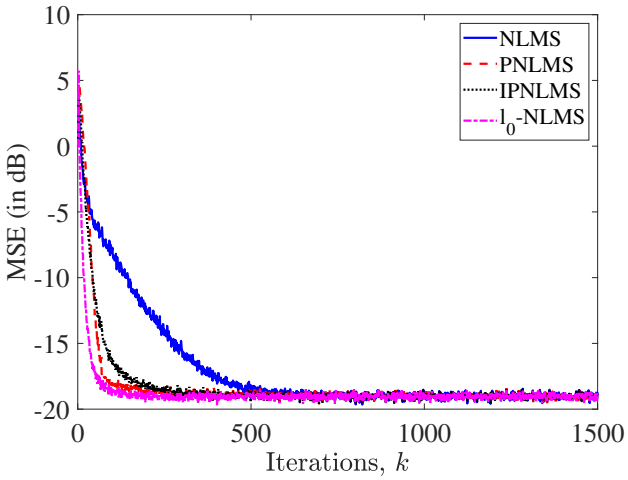
and its version shifted by 60 time samples

$$\mathbf{w}_{\mathrm{o}}^{(9)}(n) = \begin{cases} 0, & \text{if } 0 \leq n \leq 60 \ , \\ 1, & \text{if } 61 \leq n \leq 63 \ . \end{cases} \tag{3.7}$$

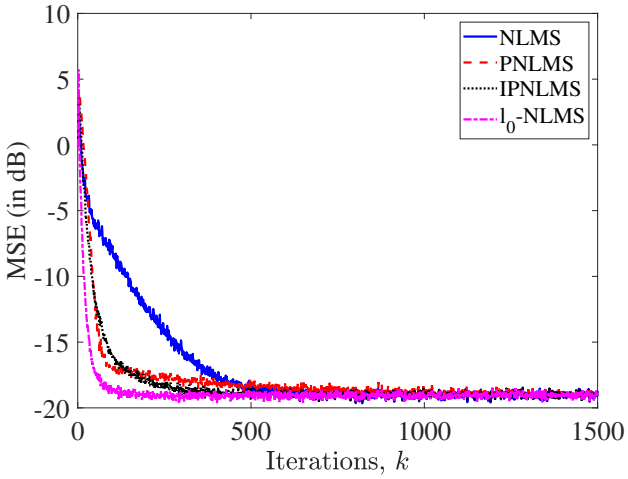
As before, we aim at compare the difference between a bad algorithm initialization and a time-shift of the impulse response, considering a sparser system. Figure 3.19 depicts the MSE learning curves of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(8)}$ and $\mathbf{w}_{\mathrm{o}}^{(9)}$. These are scenarios where the PNLMS algorithm should perform better than the other algorithms, as the impulse response is sparser than the previous example.

By initializing the algorithms with the vector $\mathbf{w}(0) = [0 \dots 0]^{\mathrm{T}}$, the PNLMS, IPNLMS and ℓ_0-NLMS algorithms converge quickly to the steady-state MSE. Figure 3.19a shows the performance of each algorithm when they initialize with this vector. With this setup, the PNLMS algorithm converges as quick as the ℓ_0-NLMS algorithm, and both are considerably faster than the IPNLMS algorithm. The NLMS algorithm takes several iterations (about 700) to reach the steady-state MSE, as expected for an algorithm that does not take any advantage of the impulse response shape.

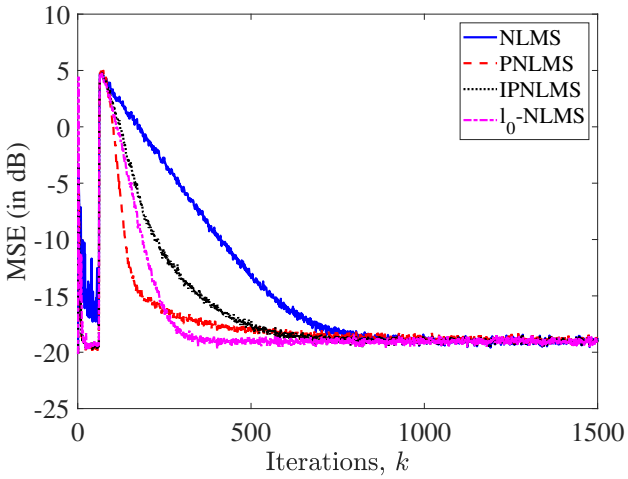
In Figure 3.19b, one can notice the worsening of the PNLMS algorithm performance as we change the initialization vector to $\mathbf{w}(0) = [0.01 \dots 0.01]^{\mathrm{T}}$. The PNLMS algorithm takes more iterations than the NLMS algorithm, even considering the



(a) Considering $\mathbf{w}_{\mathrm{o}}^{(8)}$ with initialization vector $\mathbf{w}(0) = [0 \dots 0]^{\mathrm{T}}$.



(b) Considering $\mathbf{w}_{\mathrm{o}}^{(8)}$ with initialization vector $\mathbf{w}(0) = [0.01 \dots 0.01]^{\mathrm{T}}$.



(c) Considering $\mathbf{w}_{\mathrm{o}}^{(9)}$ with initialization vector $\mathbf{w}(0) = [0 \dots 0]^{\mathrm{T}}$.

Figure 3.19: MSE learning curves of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(8)}$ and its version shifted in time $\mathbf{w}_{\mathrm{o}}^{(9)}$.

sparser impulse response in (3.6). Same as in Figure 3.18b, the IPNLMS algorithm seems more robust to this initialization. The NLMS and ℓ_0-NLMS algorithms do not change their performance at all.

Figure 3.19c shows the MSE learning curves of the algorithms considering the time-shift impulse response in (3.7). As in Figure 3.19b, the NLMS, and IPNLMS algorithms slightly outperform the PNLMS algorithm. Once more, despite the iterations that the algorithms take to fill the tapped delay line, the performance of the proportionate algorithms is very similar to the simulation that we initialize with a bad vector. One can notice that the shape of the IPNLMS algorithm learning curve is more similar to the result that the NLMS algorithm presents.

3.3.3 Section Remark

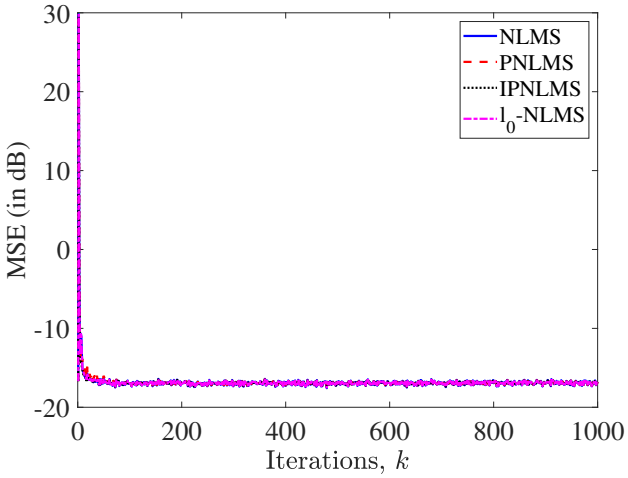
Shifting the relevant coefficients of an impulse response can reduce the convergence speed of the proportionate-type algorithms. In the beginning, the adaptive filter does unnecessary coefficient updates caused by the noise present in the desired signal. When $d'(k) \neq 0$, the adaptive filter has $\mathbf{w}(k) \neq \mathbf{w}(0)$. Thus, the new coefficients are probably far from zeroes, disturbing the good initialization of the low-magnitude coefficients.

3.4 The PNLMS Algorithm Beyond the Sparsity

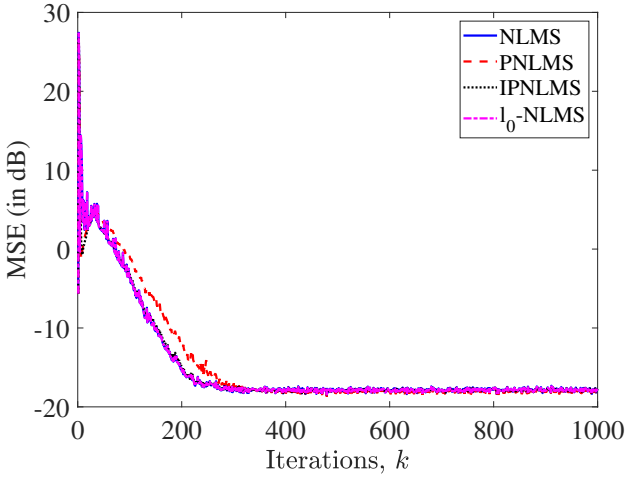
So far, we present some properties of the PNLMS algorithm that differ it from other adaptive filtering algorithms. One can interpret these properties as limitations of the PNLMS algorithm. Besides the attention that one has to take by choosing proper parameters when using the PNLMS algorithm, those same properties address useful applicability of the algorithm. As previously discussed, a proportionate-type algorithm does not rely on any tool to exploit the sparsity of an impulse response. Therefore, the sparsity of a system should not limit the performance of the PNLMS algorithm. Indeed, we can apply the PNLMS algorithm to a non-sparse system and achieve a performance as good as the one of other algorithms, by choosing suitable parameters.

In [27], the implementation of the proportionate idea does not take any advantage of the number of zeroes in the impulse response. It is clear that the PNLMS algorithm takes advantage of the large difference between the high-magnitude coefficient and the low-magnitude coefficients, besides the usual manner that one initializes an adaptive filter. However, when the initialization vector is wisely chosen, it is sufficient to make the PNLMS algorithm quickly converge to the steady-state MSE and achieving the same performance as the classical algorithms. It is worth mentioning

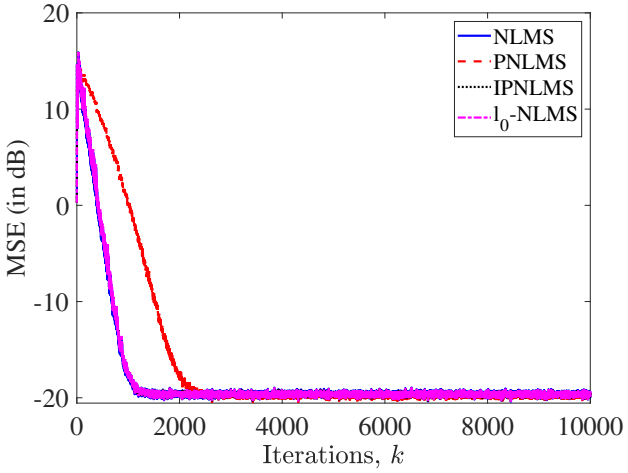
that, in practice, we need prior knowledge about the system impulse response for this application.



(a) Initialization vector $\mathbf{w}(0) = [0.9 \ldots 0.9]^{\mathrm{T}}$.



(b) Initialization vector $\mathbf{w}(0) = [0.5 \ldots 0.5]^{\mathrm{T}}$.



(c) Initialization vector $\mathbf{w}(0) = [0 \ldots 0]^{\mathrm{T}}$.

Figure 3.20: MSE learning curves of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(10)}$ for different initialization vectors.

3.4.1 First Experiment: A non-sparse impulse response with small difference among the coefficients

Suppose we want to identify the following impulse response

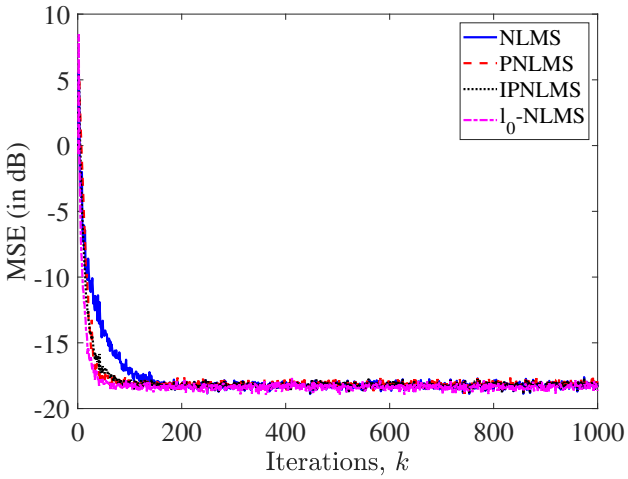
$$\mathbf{w}_{\mathrm{o}}^{(10)}(n) = \begin{cases} 1, & \text{if } n = 0,\ 1\ , \\ 0.9, & \text{if } 2 \leq n \leq 31\ . \end{cases} \tag{3.8}$$

One can notice that the impulse response $\mathbf{w}_{\mathrm{o}}^{(10)}$ is very dispersive. The system has 32 coefficients, where their magnitudes are very close. Hence, $\mathbf{w}_{\mathrm{o}}^{(10)}$ represents a non-sparse impulse response. However, by properly choosing the initial vector, the PNLMS algorithm can identify the system taking as many iterations as the other algorithms that we used so far. We perform three different simulations with the impulse response $\mathbf{w}_{\mathrm{o}}^{(10)}$. In each simulation, we initialize the algorithms with different initialization vectors: $\mathbf{w}(0) = [0.9 \ldots 0.9]^{\mathrm{T}}$, $\mathbf{w}(0) = [0.5 \ldots 0.5]^{\mathrm{T}}$ and $\mathbf{w}(0) = [0 \ldots 0]^{\mathrm{T}}$. We utilize the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms to show the importance of the initialization for the proportionate-type algorithms, in non-sparse impulse responses.

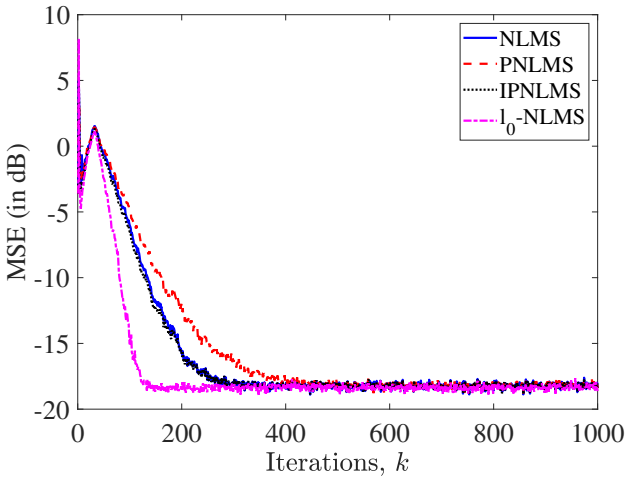
The order of the system is 31, i.e., it has 32 coefficients. All the parameters are the same as for the previous simulations. We set the step sizes aiming at evaluating all algorithms on the same MSE threshold in the steady-state. The MSE learning curves of the algorithms are computed by averaging the outcomes of 1000 independent trials.

In Figure 3.20, we can notice the learning curves for different initialization vectors. Figure 3.20b depicts the MSE learning curves of the algorithms when they initialize with the vector $\mathbf{w}(0) = [0.9 \ldots 0.9]^{\mathrm{T}}$, we set the step size $\mu = 0.99$ for all algorithms. One can notice that the performance is the same for all algorithms. Despite the sparsity of the system, the PNLMS algorithm achieves performance as good as the NLMS, IPNLMS and ℓ_0-NLMS algorithms. As we alter the initialization to a vector far from the optimum coefficients, the PNLMS algorithm performance decreases, as Figure 3.20b presents. In Figure 3.20b, we initialize with the vector $\mathbf{w}(0) = [0.5 \ldots 0.5]^{\mathrm{T}}$ and setting the step sizes for the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms to 0.75, 0.7, 0.75, and 0.75, respectively. In Figure 3.20c, we initialize with the usual vector and we set the step sizes for the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms to 0.15, 0.15, 0.1, and 0.15, respectively. By initializing the adaptive filtering algorithms with the usual vector $\mathbf{w}(0) = [0 \ldots 0]^{\mathrm{T}}$, the PNLMS algorithm takes several iterations to converge to the steady-state MSE. In this case, the IPNLMS algorithm is slightly more robust to bad initialization than the PNLMS algorithm. One must remind that the weighting matrix of the IPNLMS

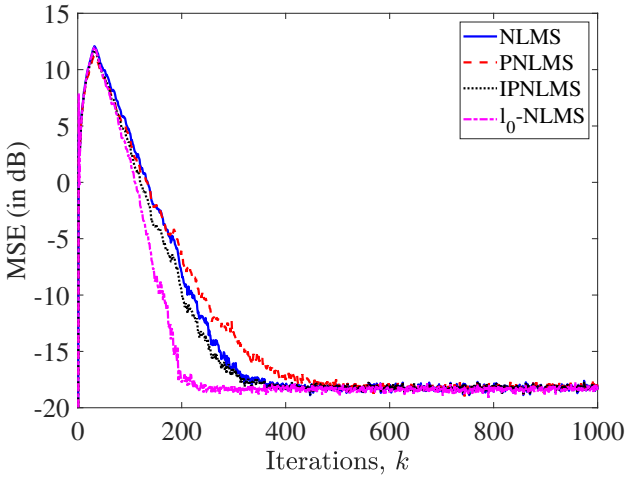
algorithm has an *NLMS* term, whose purpose is to increase the algorithm performance in dispersive systems. As the system does not have a sparse impulse response, the ℓ_0-NLMS algorithm has the same performance as the NLMS algorithm.



(a) Initialization vector $\mathbf{w}(0) = [0.01 \ldots 0.01]^{\mathrm{T}}$.



(b) Initialization vector $\mathbf{w}(0) = [0.3 \ldots 0.3]^{\mathrm{T}}$.



(c) Initialization vector $\mathbf{w}(0) = [1 \ldots 1]^{\mathrm{T}}$.

Figure 3.21: MSE learning curves of the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms considering $\mathbf{w}_{\mathrm{o}}^{(11)}$ for different initialization vectors.

3.4.2 Second Experiment: A non-sparse impulse response with large difference among the coefficients

In this experiment, we are considering a non-sparse impulse response when the magnitude of the coefficients has a large difference. Suppose that the new unknown impulse response is

$$\mathbf{w}_{\mathrm{o}}^{(11)}(n) = \begin{cases} 1, & \text{if } n = 0,\ 1\ , \\ 0.01, & \text{if } 2 \leq n \leq 32\ . \end{cases} \tag{3.9}$$

As in the previous experiment, we aim at evaluating the performance of the proportionate idea when applied to dispersive scenarios. The impulse response $\mathbf{w}_{\mathrm{o}}^{(11)}$ is dispersive, however, the difference among the magnitude of the most relevant coefficients and the other ones is larger than in the previous simulations.

The order of the system is 31, i.e., it has 32 coefficients. All the parameters are the same as for the previous simulations. We set the step sizes for the NLMS, PNLMS, IPNLMS, and ℓ_0-NLMS algorithms to 0.65, 0.5, 0.65, and 0.99, respectively. The MSE learning curves of the algorithms are computed by averaging the outcomes of 1000 independent trials.

Figure 3.21 depicts the MSE learning curves of the algorithms for different initialization vectors. One can notice in Figure 3.21a, by initializing the algorithms with vector $\mathbf{w}(0) = [0.01 \ldots 0.01]^{\mathrm{T}}$ that the PNLMS and IPNLMS algorithms require few iterations to converge to the steady-state MSE. In this simulation, the NLMS algorithm has the worst performance. However, as we initialize the algorithms with vector far from the optimum value of the low-magnitude coefficients, the performance of the proportionate algorithms are impaired, as presented in Figures 3.21b and 3.21c. One can notice that this bad initialization make the PNLMS algorithm to perform worse than the NLMS algorithm, even considering an impulse response with a large difference among the magnitude of the coefficients.

3.4.3 Section Remark

Independently of the system type, sparse or dispersive, proportionate-type algorithms rely on good initialization to achieve better results. The initialization vector should be close to the low-magnitude coefficients. The choice of the initialization vector is more critical to the PNLMS algorithm.

Chapter 4

The Simple Sparsity-Aware Feature LMS Algorithm

In the previous chapter, we address some properties of the proportionate algorithms, concluding that they do not exploit sparsity, only taking advantage of some features of the systems. Recently, it has been verified that many systems have some type of sparsity, be it *plain* or *hidden*. As said before, the plain sparsity occurs when most coefficients of the system has low magnitude, i.e., sparsity is directly observed in the current representation of the system. On the other hand, when some mathematical manipulation is required to reveal the system sparsity, we say that this system has hidden sparsity. The LMS algorithm is usually a good option in several scenarios for its simplicity and low complexity. Unfortunately, the LMS algorithm does not take advantage of any type of sparsity.

The problem of plain sparsity was addressed for a while, and there exist many algorithms that take advantage in some manner of it [27, 39, 40, 47, 54, 69–71]. Many works have verified that plain sparsity can be best represented by the ℓ_0-norm [51, 72, 73], and that is the main idea behind the ℓ_0-norm LMS algorithm [53]. In comparison to the proportionate algorithms, the ℓ_0-norm LMS algorithm presents better performance in sparse scenarios, as discussed in Chapter 3. Differently from these algorithms, the recently proposed feature LMS (F-LMS) algorithm [13] can benefit from the hidden sparsity by exploiting some features inherent to the unknown system.[1]

Naturally, there are systems that have both plain and hidden sparsity. However, the F-LMS and ℓ_0-norm LMS algorithms can not exploit both types of sparsity simultaneously. Hence, by imposing plain sparsity promoting functions to the cost

[1] While the F-LMS algorithm exploits features like lowpass or highpass spectrum through linear combinations of coefficients, there are other features that could not be exploited in the same manner. For example, the tensor LMS algorithm exploits impulse responses that can be decomposed as the Kronecker product of two lower-dimensional impulse responses [55].

function of the F-LMS algorithm, we have a new algorithm that is able to improve its performance by exploiting both plain and hidden sparsity at the same time. Therefore, we propose to include a trivial penalty function on the adaptive coefficients in the cost function of the F-LMS algorithm. This penalty function relies on the so-called *discard function* [56], then the simple sparsity-aware feature LMS (SSF-LMS) algorithm reduces its number of arithmetic operations, thus saving important computational resources, and being able to outperform the F-LMS algorithm when the system has both types of sparsity.[2]

4.1 The F-LMS Algorithm using ℓ_1-norm

Several research efforts have been made in the adaptive filtering field that aims to exploit some sparsity in the coefficients by imposing any type of constraint in a cost function [51, 56, 74–77]. However, these works refer only to the plain sparsity, by allowing the attraction of some coefficients to zero. A new family of algorithms was proposed in [13], where the sparsity arises from the linear combination of adaptive coefficients of the filter. The F-LMS family of algorithms induces some sparsity properties hidden in the coefficients. Thus, these algorithms require a feature matrix $\mathbf{F}(k)$ that determines the feature we seek in the system [13]. The F-LMS algorithm minimizes the following cost function

$$\xi_{\text{F-LMS}}(k) = \frac{1}{2}|e(k)|^2 + \alpha\mathcal{P}\left(\mathbf{F}(k)\mathbf{w}(k)\right) \ , \tag{4.1}$$

where $\alpha \in \mathbb{R}_+$ is the weight given to the sparsity-promote penalty function $\mathcal{P}$, and $\mathbf{F}(k)$ is the *feature matrix* capable of exploiting the features inherent to the unknown system. This matrix is responsible for revealing the hidden sparsity, i.e., by applying $\mathbf{F}(k)$ to $\mathbf{w}(k)$ we perform a linear combination that intends to reveal a sparse vector (meaning that the vector $\mathbf{F}(k)\mathbf{w}(k)$ should have most of its entries equal or close to zero). In practice, we should have some prior knowledge about the system before we choose the feature matrix. However, for many real systems a suitable feature is easy to identify, for example, many analog systems exhibit lowpass feature due to the use of high sampling rates. The matrix $\mathbf{F}(k)$ can vary at each iteration, representing a more general feature or to track a time-varying unknown system, for simplicity we focus on simple algorithms, thus we assume the feature matrix as time-invariant $\mathbf{F}(k) = \mathbf{F}$ as in [13].

The penalty function $\mathcal{P}$ in (4.1) can be any almost everywhere differentiable sparsity-promoting function to allow for gradient-based methods. Many functions

[2]Recently, an alternative approach based on the ℓ_0-norm has been proposed in [57], but the paper proposes an algorithm that requires a larger number of arithmetic operations in comparison with the SSF-LMS algorithm.

satisfy this condition: vector norms [74, 77]; vector norms combined with shrinkage strategies [56]; functions that approximate the ℓ_0-norm [51, 72]; and others [13]. We choose the ℓ_1-norm as our penalty function $\mathcal{P}$ like in [13], thus the complexity of the F-LMS algorithm is only slightly superior to the LMS algorithm complexity. Therefore, the resulting objective function is

$$\xi_{\text{F-LMS}}(k) = \frac{1}{2}|e(k)|^2 + \alpha\|\mathbf{F}\mathbf{w}(k)\|_1 \ , \tag{4.2}$$

and the general update equation is given by

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k) - \mu\alpha\mathbf{p}(k) \ , \tag{4.3}$$

where $\mu \in \mathbb{R}_+$ is the step size, which should be small enough to ensure convergence [1], and $\mathbf{p}(k) \in \mathbb{R}^{N+1}$ is the gradient of function $\|\mathbf{F}\mathbf{w}(k)\|_1$ with respect to the adaptive coefficients $\mathbf{w}(k)$.

The complete operation of the general F-LMS algorithm is given in Algorithm 7.

Algorithm 7: The F-LMS using ℓ_1-norm

Initialization:

$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$

choose μ in the range $0 < \mu \ll 1$

choose α in the range $0 < \alpha < 1$

Do for $k \geq 0$

$e(k) = d(k) - \mathbf{w}^T(k)\mathbf{x}(k)$

Compute $\mathbf{p}(k)$, refer to (4.5) and (4.8) for example

$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k) - \mu\alpha\mathbf{p}(k)$

In the following subsections, we describe two simple versions as examples of the F-LMS algorithm exploiting the lowpass and highpass features of the unknown systems. Remembering that the F-LMS algorithm exploits more complex features, but we intend to present simple algorithms, also an easy and straightforward manner to illustrate how a feature can be exploited.

4.1.1 The F-LMS algorithm for lowpass systems

Several systems concentrate most of their energy in low frequency components. The main behavior of this type of systems is its lowpass narrowband spectrum, resulting in a smooth impulse response. Therefore, if the system has lowpass narrowband spectrum, then the magnitude of adjacent coefficients are close to each other, i.e., the difference between adjacent coefficients of the impulse response $\mathbf{w}_o$ is small. By properly choosing the feature matrix, we can impose a new condition to the cost function and minimize the sum of two adjacent coefficients. In this case, we set $\mathbf{F}$

as $\mathbf{F}_l$, where $\mathbf{F}_l$ is an $\mathrm{N} \times \mathrm{N}+1$ matrix defined as

$$\mathbf{F}_l = \begin{bmatrix} 1 & -1 & 0 & \cdots & 0 \\ 0 & 1 & -1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \\ 0 & 0 & \cdots & 1 & -1 \end{bmatrix} \tag{4.4}$$

and $\|\mathbf{F}_l \mathbf{w}(k)\|_1 = \sum_{i=0}^{\mathrm{N}-1} |w_i(k) - w_{i+1}(k)|$. Therefore, the F-LMS algorithm for lowpass systems is defined by the general update equation given in (4.3), but replacing the vector $\mathbf{p}(k)$ with the gradient for lowpass systems $\mathbf{p}_l(k)$ whose entries are given by

$$p_{l,i}(k) = \begin{cases} \mathrm{sgn}(w_0(k) - w_1(k)), & \text{if } i = 0 \ , \\ -\mathrm{sgn}(w_{i-1}(k) - w_i(k)) + \mathrm{sgn}(w_i(k) - w_{i+1}(k)), & \text{if } i = 1, \cdots, \mathrm{N}-1 \ , \\ -\mathrm{sgn}(w_{N-1}(k) - w_N(k)), & \text{if } i = \mathrm{N} \ . \end{cases} \tag{4.5}$$

where

$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x > 0 \ , \\ 0, & \text{if } x = 0 \ , \\ -1, & \text{if } x < 0 \ . \end{cases} \tag{4.6}$$

4.1.2 The F-LMS algorithm for highpass systems

The counterpart of the lowpass narrowband spectrum is the highpass narrowband spectrum. Differently from the previous case, a system has highpass narrowband spectrum when it is mainly defined by its high frequency components. Hence, adjacent coefficients of the impulse response of such system vary quickly, although in some predictable manner, i.e., adjacent coefficients have similar absolute values with opposite signs if the system has highpass narrowband spectrum. Therefore, we aim to minimize the sum of adjacent adaptive coefficients $\mathbf{w}(k)$ since the sum of two consecutive coefficients is close to zero, exploiting the feature of the highpass system. We can accomplish this by selecting $\mathbf{F}$ as $\mathbf{F}_h$, where $\mathbf{F}_h$ is an $\mathrm{N} \times \mathrm{N}+1$ feature matrix defined as

$$\mathbf{F}_h = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 0 \\ \vdots & & \ddots & \ddots & \\ 0 & 0 & \cdots & 1 & 1 \end{bmatrix} \ , \tag{4.7}$$

such that $\|\mathbf{F}_h\mathbf{w}(k)\|_1 = \sum_{i=0}^{N-1} |w_i(k)+w_{i+1}(k)|$. Similarly to the case of the lowpass filter, we can characterize the F-LMS algorithm for highpass systems by the general update equation given in (4.3), but replacing $\mathbf{p}(k)$ with $\mathbf{p}_h(k)$, whose entries are given by

$$p_{h,i}(k) = \begin{cases} \text{sgn}(w_0(k) + w_1(k)), & \text{if } i = 0 \ , \\ \text{sgn}(w_{i-1}(k) + w_i(k)) + \text{sgn}(w_i(k) + w_{i+1}(k)), & \text{if } i = 1, \cdots, \text{N} - 1 \ , \\ \text{sgn}(w_{N-1}(k) + w_N(k)), & \text{if } i = \text{N} \ . \end{cases} \tag{4.8}$$

4.2 The SSF-LMS Algorithm

The F-LMS algorithms do some linear combination to exploit the hidden sparsity in the parameters of the unknown system, i.e., the sparsity is revealed by applying the feature matrix $\mathbf{F}$ to $\mathbf{w}(k)$. However, there are many cases in which there exists plain sparsity in the parameters, i.e., $\mathbf{w}_o$ already represents a sparse vector. Such impulse response does not have a feature for the F-LMS algorithms to take advantage. Thus, these algorithms can not exploit the plain sparsity. In this work, we propose an algorithm that relies on another technique in addition to the features of the system, then it can exploit both types of sparsity simultaneously. In combination to the feature matrix, we also impose a constraint on the adaptive coefficients aiming their attraction to zero. The SSF-LMS algorithm minimizes the following objective function

$$\xi(k) = \frac{1}{2}|e(k)|^2 + \alpha\|\mathbf{F}\left[\mathbf{f}_\epsilon\left(\mathbf{w}(k)\right)\right]\|_1 \ , \tag{4.9}$$

where $\mathbf{f}_\epsilon(\mathbf{w}(k)) = [f_\epsilon(w_0(k)) \ f_\epsilon(w_1(k)) \ldots f_\epsilon(w_{\text{N}}(k))]^{\text{T}}$ is the discard function, whose i^{th} element is defined as [56]

$$f_\epsilon(w_i(k)) = \begin{cases} w_i(k), & \text{if } |w_i(k)| \geq \epsilon \ , \\ 0, & \text{if } |w_i(k)| < \epsilon \ , \end{cases} \tag{4.10}$$

the parameter $\epsilon \in \mathbb{R}_+$ is a threshold chosen by the user, generally close to the measurement noise [56]. In comparison to (4.2), the objective function in (4.9) generates the following update equation

$$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k) - \mu\alpha\mathbf{p}(k)\mathbf{J}_{\mathbf{f}_\epsilon(\mathbf{w}(k))} \ , \tag{4.11}$$

where $\mathbf{J}_{\mathbf{f}_\epsilon(\mathbf{w}(k))}$ is a diagonal matrix whose diagonal elements are defined as

$$\mathbf{J}_{\mathbf{f}_\epsilon(\mathbf{w}(k))i,i} = \begin{cases} 1, & \text{if } |w_i(k)| \geq \epsilon \ , \\ 0, & \text{if } |w_i(k)| < \epsilon \ . \end{cases} \tag{4.12}$$

Therefore, matrix $\mathbf{J}_{\mathbf{f}_\epsilon(\mathbf{w}(k))}$ just selects the entries of vector $\mathbf{p}(k)$ which are relevant and, as a consequence, one can implement (4.11) efficiently by not computing the entries of $\mathbf{p}(k)$ related to the coefficients with small magnitude (plain sparsity) detected by (4.12). Indeed, this technique can greatly reduces the number of arithmetic operations saving even more computational resources.

Algorithm 8: The SSF-LMS algorithm

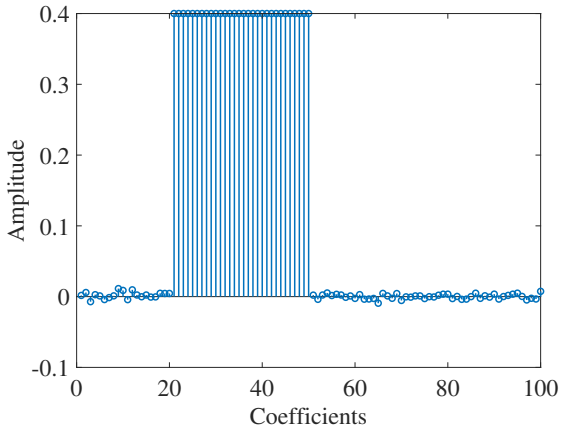
Initialization:
$\mathbf{x}(0) = \mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$
choose μ in the range $0 < \mu \ll 1$
choose α in the range $0 < \alpha < 1$
choose ϵ small, close to measurement error
Do for $k \geq 0$
$e(k) = d(k) - \mathbf{w}^T(k)\mathbf{x}(k)$
$\mathbf{p}(k) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$
Do for $i = 0$ to N
if $|w_i(k)| > \epsilon$
Compute $p_i(k)$
$\mathbf{w}(k+1) = \mathbf{w}(k) + \mu e(k)\mathbf{x}(k) - \mu\alpha\mathbf{p}(k)$

The SSF-LMS algorithm is summarized in Algorithm 8. Vector $\mathbf{p}(k)$ in (4.11) is the gradient of the penalty function and the feature matrix, i.e., the same as those shown in Section 4.1, for lowpass and highpass systems. One can notice that the new step introduced by the SSF-LMS algorithm (the *if loop* in Algorithm 8) reduces the computation required for $p(k)$, i.e., reduces the number of multiplication and additions in each iteration.

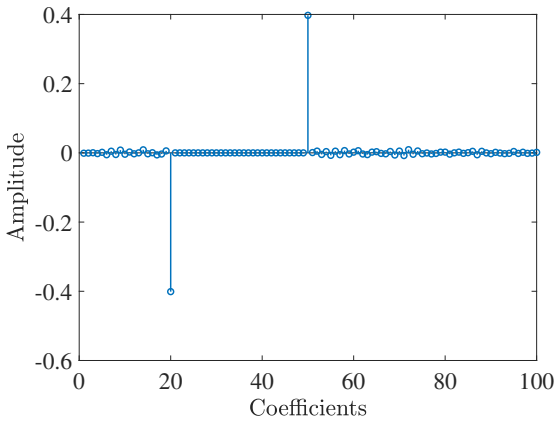
4.3 Simulations and Experiments Description

In this section, we present some impulse responses that have both plain and hidden sparsity, also a time-varying impulse response. The experiments aim at verifying the potential benefits of exploiting both types of sparsity simultaneously, by applying several LMS-based algorithms to identify some unknown sparse lowpass and sparse highpass systems. The competing algorithms are: (i) the LMS algorithm; (ii) the F-LMS algorithm, which can exploit only hidden sparsity; and (iii) the ℓ_0-LMS algorithm, which is able to exploit only plain sparsity and achieves better results

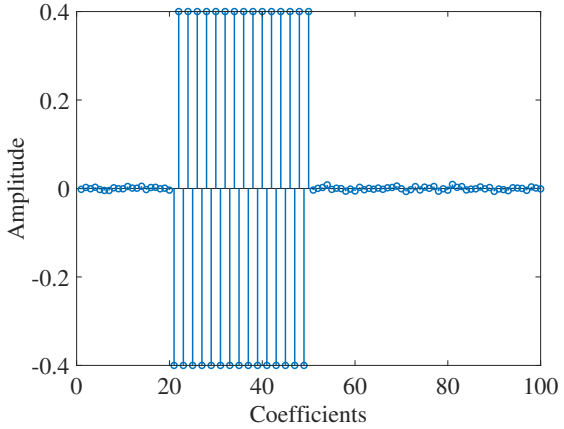
than most sparsity-aware LMS-based algorithms, then constituting a benchmark among them [51, 53].



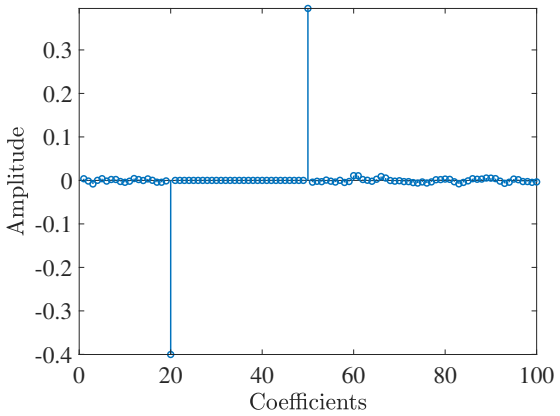
(a) Impulse response of the first simulation $\mathbf{w}_{o,l}$.



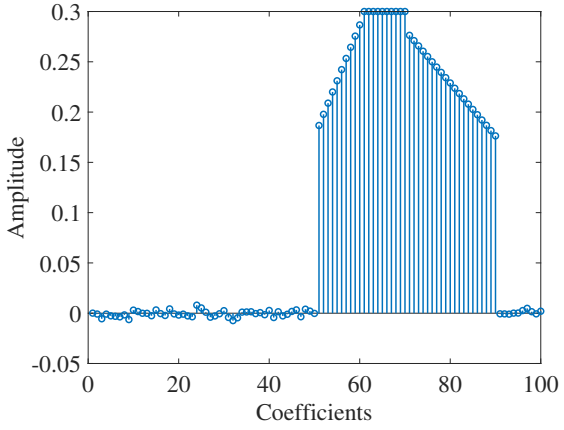
(b) Impulse response of the first simulation $\mathbf{w}_{o,l}$ in the transformed domain.



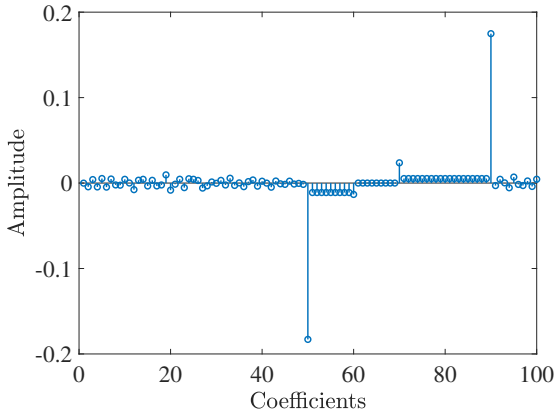
(c) Impulse response of the second simulation $\mathbf{w}_{o,h}$.



(d) Impulse response of the second simulation $\mathbf{w}_{o,h}$ in the transformed domain.



(e) Impulse response of the third simulation $\mathbf{w}'_{o,l}$ after 2000 iterations.



(f) Impulse response of the third simulation $\mathbf{w}'_{o,l}$ in the transformed domain.

Figure 4.1: Impulse response of the unknown systems.

The order of all unknown systems is 99, i.e., they have 100 coefficients, among which 30 to 40 are considered relevant (that is, their magnitudes are much greater

than the magnitude of other coefficients, as illustrated in Fig. 4.1). The input signal is a zero-mean white Gaussian noise, which has unit variance. The signal-to-noise ratio (SNR) is chosen as 20 dB. All algorithms have as initialization vector $\mathbf{w}(0) = [0 \cdots 0]^{\mathrm{T}}$, moreover [3] $\alpha = 0.05$, $\sigma_m^2 = 10^{-5}$ and $\epsilon = 10^{-2}$. We use the Laplacian form with first order truncation for the ℓ_0-LMS, as in Section 2.5, and we set the weight given to the approximation of the ℓ_0-norm as $\kappa = 2 \times 10^{-3}$ and the parameter that controls the quality of the ℓ_0-norm approximation is $\beta = 5$ [51, 53]. For the sake of clarity, the values of the step size μ are informed later for each simulation result. The MSE learning curves of the presented algorithms are computed averaging the outcomes of 500 independent trials.

In the first experiment, we seek to compare the performance of the aforementioned algorithms when the unknown systems stand still, i.e., do not change along the iterations. For this experiment, we consider two impulse responses. The first unknown system, $\mathbf{w}_{\mathrm{o,l}}$, is a block sparse lowpass system whose first 20 coefficients are zero-mean white Gaussian with variance σ_m^2 (these coefficients represent the plain sparsity of this system), the next 30 coefficients are constant and equal to 0.4 (these coefficients refer to the low frequency components) and the last 50 coefficients are also zero-mean white Gaussian with variance σ_m^2 (these coefficients also represent a block of plain sparsity). Fig. 4.1a depicts such impulse response. The second unknown system, $\mathbf{w}_{\mathrm{o,h}}$, is almost the same as the first one, but the non-white Gaussian coefficients (relevant coefficients) with odd and even indexes are -0.4 and 0.4, respectively. We illustrate this system impulse response in Fig. 4.1c, which is the highpass filter.

In the experiment experiment, we use a time-variant impulse response to test the tracking capability of the SSF-LMS algorithm in comparison to the others algorithms. The time-variant unknown system starts as the same block sparse lowpass system $\mathbf{w}_{\mathrm{o,l}}$ used in the simulations of first experiment. However, after 2000 iterations, the system coefficients change to

$$\mathbf{w}'_{\mathrm{o,l}}(n) = \begin{cases} \dfrac{0.1n}{9} - 0.38, & \text{if } 51 \leq n < 61 \ , \\ 0.3, & \text{if } 61 \leq n < 71 \ , \\ -\dfrac{0.1n}{19} + 0.65, & \text{if } 71 \leq n < 91 \ , \end{cases}$$

where n is the coefficient index. The other coefficients (that represent the plain sparse portion of the impulse response) are zero-mean white Gaussian with variance σ_m^2. All of the other parameters are the same as in the first experiment. Fig. 4.1e depicts such system impulse response. One can notice that $\mathbf{w}'_{\mathrm{o,l}}$ continues to be

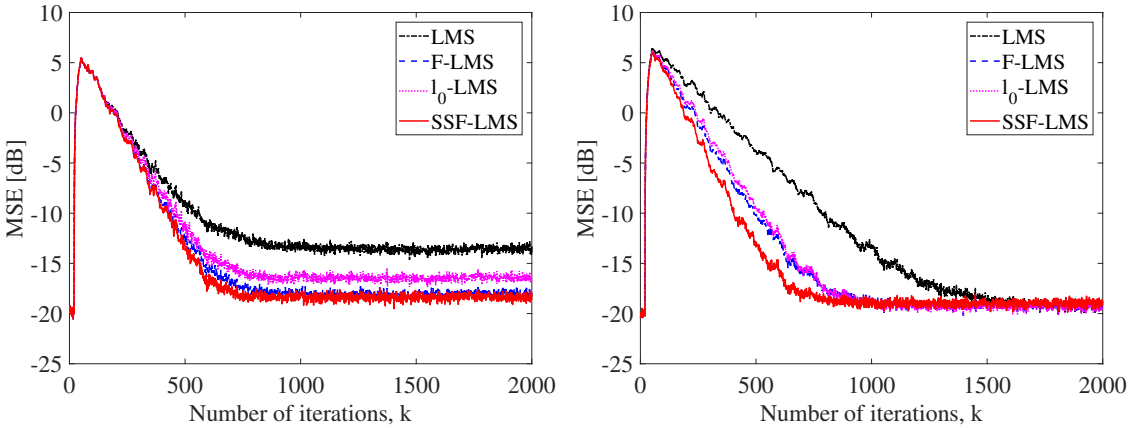
[3] The variance σ_m^2 refers to the noise inserted in the low-magnitude coefficients of the unknown systems. Therefore, it shows the robustness of the SSF-LMS algorithm in compressive scenarios.

a lowpass impulse response, but now, such feature appears in blocks. Instead of a single block where the coefficients are the same, now we have blocks where the magnitude of adjacent coefficients slowly vary, i.e., adjacent coefficients are similar in determined regions of the impulse response $\mathbf{w}'_{o,l}$.

4.4 Discussion of Results

In this section, we present the MSE learning curves of the tested algorithms as result for the simulations and experiments depicted in the previous section. We also discuss this results by comparing the performance in the steady-state MSE and convergence speed, of each algorithm. Additionally, we show the number of arithmetic operations per iteration in the steady-state that each algorithm requires to identify each impulse response.

Fig. 4.2 depicts the MSE learning curves of the LMS, F-LMS, ℓ_0-LMS, and SSF-LMS algorithms using the block sparse lowpass impulse response $\mathbf{w}_{o,l}$ in two different simulations scenarios. In Fig. 4.2a, all algorithms use the same step size $\mu = 0.015$, therefore they show similar convergence speeds. This analysis compares the steady-state MSE that each algorithm is able to reach, verifying their precision. We can notice that the SSF-LMS algorithm achieves the lowest MSE, followed by the F-LMS, ℓ_0-LMS and LMS algorithms, respectively. Although one can notice the MSE results of the SSF-LMS algorithm are only slightly superior, in relation to the MSE reached by the F-LMS algorithm, one must remind that the SSF-LMS algorithm is a low complexity solution, which performs fewer arithmetic operations due to the plain sparsity presented in $\mathbf{w}_{o,l}$.



(a) All algorithms with the same step size: $\mu = 0.015$.

(b) LMS, F-LMS, ℓ_0-LMS, and SSF-LMS algorithms with step sizes equal to 0.003, 0.0055, 0.005, and 0.007, respectively.

Figure 4.2: MSE learning curves of the LMS, F-LMS, ℓ_0-LMS, and SSF-LMS algorithms considering $\mathbf{w}_{o,l}$.

In Fig. 4.2b, we compare the algorithms convergence speed by letting fix the steady-state MSE. Hence, we alter the step sizes for the LMS, F-LMS, ℓ_0-LMS, and SSF-LMS algorithms to 0.003, 0.0055, 0.005, and 0.007, respectively. This analysis provides a different view about the capacity of the algorithms, mainly the proposed SSF-LMS algorithm. One can notice that the SSF-LMS algorithm converges to the steady-state MSE much faster than the others algorithms. It is worthy to mention that the SSF-LMS algorithm reaches these results by performing fewer arithmetic operations than its competitors, due to the existing plain sparsity of the lowpass system. Hence, the SSF-LMS algorithm outperforms the F-LMS, ℓ_0-LMS and LMS algorithms, for lowpass systems with plain and hidden sparsity simultaneously.

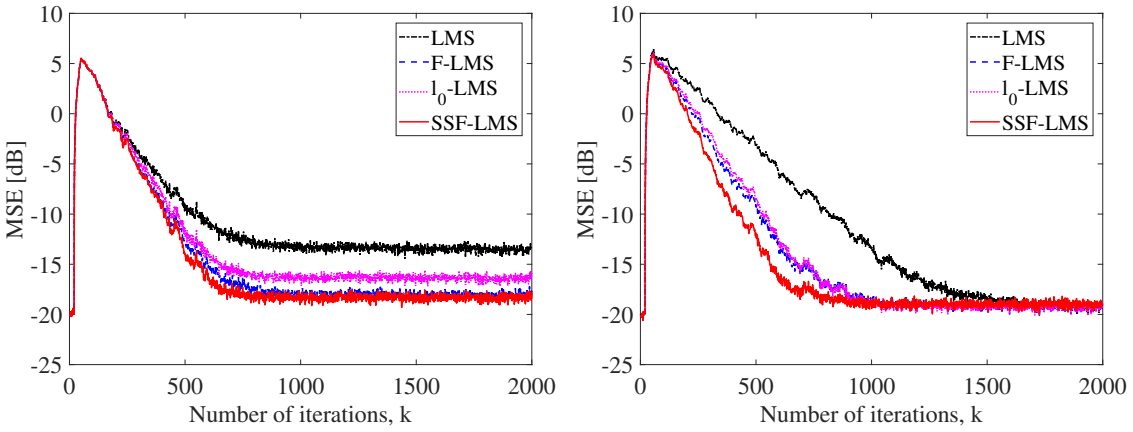
Table 4.1: Number of arithmetic operations per iteration during steady-state considering $\mathbf{w}_{\mathrm{o,l}}$.

Algorithm	# Multiplications	# Additions
SSF-LMS	232	321
F-LMS	301	497
LMS	201	200
ℓ_0-LMS	341	340

Table 4.1 depicts the number of arithmetic operations that each algorithm does during the steady-state considering the lowpass impulse response $\mathbf{w}_{\mathrm{o,l}}$. We can notice that, besides achieving better performance in both simulations, the SSF-LMS algorithm also requires fewer arithmetic operations in comparison to the F-LMS and ℓ_0-LMS algorithms. The LMS algorithm requires the fewest arithmetic operation, however it has the worst performance among the algorithms and does not exploit any type of sparsity. The reduction in computation occurs whenever there is observed plain sparsity in the unknown impulse response, since in this case the SSF-LMS algorithm does not compute every entry of vector $\mathbf{p}(k)$ (refer to Algorithm 2). The algorithm computes only those entries that are relevant for the system, i.e., those entries that are larger than the threshold forced by the discard function. Therefore, we can analyze the limiting case where all coefficients are relevant (there is no plain sparsity in the impulse response), the SSF-LMS algorithm would perform exactly the same number of arithmetic operations required by the F-LMS algorithm.

In Fig. 4.3, we show the MSE learning curves of the LMS, F-LMS, ℓ_0-LMS, and SSF-LMS algorithms, which are the results of simulations for the block sparse highpass system $\mathbf{w}_{\mathrm{o,h}}$. Fig. 4.3 show the results when the algorithms have the same convergence speed and same steady-state MSE as in Fig. 4.2. Fig. 4.3a depicts the performance of the algorithms when all of them have the same step size ($\mu = 0.015$), comparing their steady-state MSE threshold. Once again, the SSF-LMS algorithm reaches the lowest MSE but the difference between its performance and the F-LMS algorithm is not remarkable. One can notice that the SSF-LMS algorithm reaches

just few dBs less than the F-LMS algorithm. For the next simulation we set the step sizes for the LMS, F-LMS, ℓ_0-LMS, and SSF-LMS algorithms to 0.003, 0.0055, 0.005, and 0.007, respectively. We can observe in Fig. 4.3b the comparison of the convergence speed of each algorithm. As in the simulation considering $\mathbf{w}_{o,l}$, the SSF-LMS algorithm achieves the best convergence rate.



(a) All algorithms with the same step size: $\mu = 0.015$.

(b) LMS, F-LMS, ℓ_0-LMS, and SSF-LMS algorithms with step sizes equal to 0.003, 0.0055, 0.005, and 0.007, respectively.

Figure 4.3: MSE learning curves of the LMS, F-LMS, ℓ_0-LMS, and SSF-LMS algorithms considering $\mathbf{w}_{o,h}$.

By considering that the impulse response $\mathbf{w}_{o,h}$ has the same number of relevant coefficients as $\mathbf{w}_{o,l}$, the number of arithmetic operations of each algorithm during steady-state is exactly the same as those depicted in Table 4.1. In spite of this, for convenience, we rewrite this numbers in Table 4.2.

Table 4.2: Number of arithmetic operations per iteration during steady-state considering $\mathbf{w}_{o,h}$.

Algorithm	# Multiplications	# Additions
SSF-LMS	232	321
F-LMS	301	497
LMS	201	200
ℓ_0-LMS	341	340

The second experiment validates the tracking capability of the SSF-LMS algorithm, where after several iterations the impulse response completely changes. However, the system keeps its lowpass feature, thus the feature matrix does not need to be altered. For this experiment we fix the steady-state MSE of the algorithms, by setting $\mu = 0.015$. According to Fig. 4.4 we observe that the SSF-LMS algorithm continues to reach the steady-state MSE first, despite the sudden variation in the unknown impulse response.

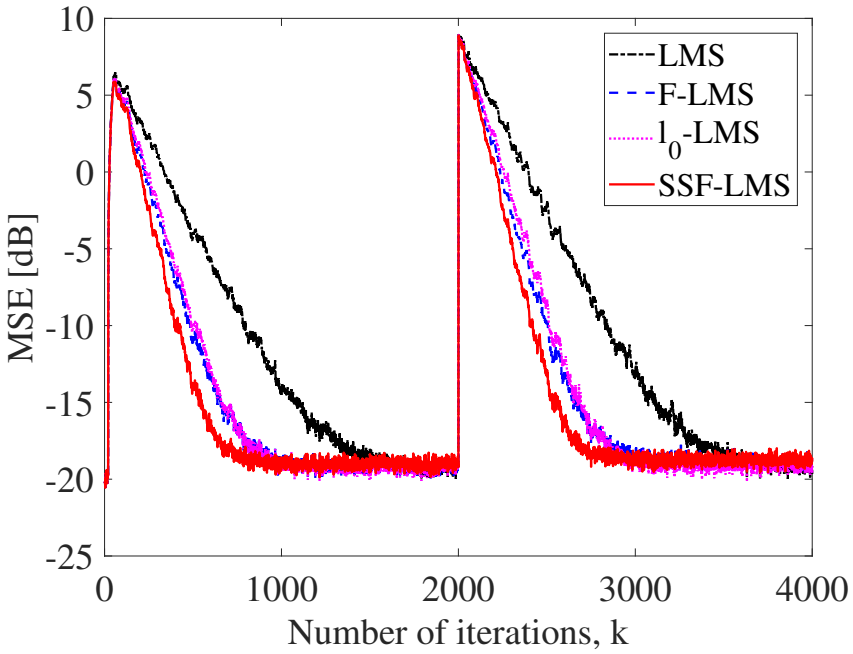


Figure 4.4: MSE learning curves of the LMS, F-LMS, ℓ_0-LMS, and SSF-LMS algorithms considering that the unknown system is $\mathbf{w}_{o,1}$ in the first 2000 iterations, and suddenly changed to $\mathbf{w}'_{o,1}$. The step sizes for each algorithm are the same as those used in Fig 4.2b.

Table 4.3: Number of arithmetic operations per iteration during steady-state considering $\mathbf{w}'_{o,1}$.

Algorithm	# Multiplications	# Additions
SSF-LMS	242	361
F-LMS	301	497
LMS	201	200
ℓ_0-LMS	321	320

Table 4.3 presents the number of arithmetic operations during steady-state considering only the new impulse response $\mathbf{w}'_{o,1}$. As expected the SSF-LMS algorithm requires lower amount of operations (in total) than the F-LMS and the ℓ_0-LMS algorithms, but, in this simulation, the SSF-LMS algorithm requires more additions per iteration than the ℓ_0-LMS algorithm. However, this is a minor problem as the SSF-LMS algorithm performs much fewer multiplications, which are operations that demand more computational power than additions.

Chapter 5

Conclusions

In this work, we did a profound study about the LMS-based algorithms, including some that are sparsity-aware. We performed a brief review of the studied algorithms, intending to expose some significant individualities of each algorithm. We show the differences between the proportionate and regularization families. We could not see the way that a proportionate algorithm can exploit the system sparsity when compared to the regularization algorithms.

We addressed some properties of the proportionate-type algorithms. We started by studying the role of the weighting matrix $\mathbf{G}(k)$ in the PNLMS algorithm operation. However, this analysis can be extended to some other algorithms of this family. We saw that a proportionate algorithm relies on a large difference between the estimated coefficients. To a high-magnitude parameter converges faster, the low-magnitude ones need to be slower. As a sparse impulse response has the most of its energy concentrated in a short time range, and one usually initializes an adaptive filter with the vector $\mathbf{w}(0) = [0\ 0\ \ldots\ 0]^{\mathrm{T}}$, the proportionate algorithms achieve excellent performance, mainly the PNLMS algorithm. However, this creates dependence in the initialization vector. The step sizes assigned to the low-magnitude coefficients are small enough to slow them to the point where it impairs the algorithm performance, thus it relies on an initialization vector close to the low-magnitude coefficients. Hence, strictly speaking, we conclude that the proportionate-type family does not exploit the system sparsity.

From these two properties, we assigned others that fortify our hypothesis. The time-shift degraded the performance of the algorithms, even in a sparse impulse response. When there is a shift in time, an adaptive filter updates its coefficients considering only the measurement noise. Until the relevant coefficient does not feed the adaptive filter, the learning process corrupts the initialization vector, creating a new one that we do not have any knowledge. However, besides all of this, we presented that, with proper parameters set, the PNLMS algorithm can achieve satisfactory performance in non-sparse systems.

We introduced a penalty function to the cost function of the F-LMS algorithm, then both types of sparsity can be exploited. The discard function exploits the plain sparsity, whereas the feature matrix unveils the hidden sparsity. We compared the performance of the proposed SSF-LMS algorithm to other sparsity-aware algorithms, in some sense. By elaborating some examples, which aimed at identifying impulse responses that have both types of sparsity, we conclude that the SSF-LMS algorithm presented the best performance in terms of the MSE, while also requiring fewer arithmetic operations than the F-LMS algorithm.

In future works, we will explain strategies that facilitate the learning process of the feature matrix. By calculating the feature online, we can eliminate the necessity for prior information about the system.

Bibliography

[1] DINIZ, P. S. R. *Adaptive Filtering: Algorithms and Practical Implementation.* 4th ed. New York, USA, Springer, 2013.

[2] DINIZ, P. S. R. *Digital Signal Processing: System Analysis and Design.* 2nd ed. Cambridge, UK, Cambridge University Press, 2012.

[3] HAYKIN, S. *Adaptive Filter Theory.* 4th ed. Englewood Cliffs, USA, Prentice Hall, 2002.

[4] ANTONIOU, A. *Digital Filters: Analysis, Design and Applications.* 2nd ed. New York, USA, McGraw Hill College, 1993.

[5] WIDROW, B., STEARNS, S. *Adaptive Signal Processing.* 1st ed. Englewood Cliffs, USA, Prentice Hall, 1985.

[6] SAYED, A. H. *Fundamentals of Adaptive Filtering.* New York, USA, Wiley-IEEE, 2003.

[7] BROWN, R. G. *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises.* 4th ed. New York, USA, Wiley, 2012.

[8] WIENER, N. *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications.* Cambridge, USA, The MIT Press, 1964.

[9] PEYTON Z., P. J. *Probability, Random Variables, and Random Signal Principles.* 4th ed. New York, USA, McGraw Hill, 2000.

[10] FELLER, W. *An Introduction to Probability Theory and Its Applications.* 3rd ed. New York, USA, Wiley, 1968.

[11] PAPOULIS, A. *Probability, Random Variables, and Stochastic Processes.* 3rd ed. New York, USA, McGraw Hill, 1991.

[12] WIDROW, B., HOFF, M. E. "Adaptive switching circuits", *IRE WESCOM Conv. Rec.*, v. 4, pp. 96–104, 1960.

[13] DINIZ, P. S. R., YAZDANPANAH, H., LIMA, M. V. S. "Feature LMS Algorithms". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4144–4148, Alberta, Canada, April 2018.

[14] UNGERBOECK, G. "Theory on the Speed of Convergence in Adaptive Equalizers for Digital Communication", *IBM Journal of Research and Development*, v. 16, n. 6, pp. 546–555, Nov 1972.

[15] MAZO, J. E. "On the Independence Theory of Equalizer Convergence", *The Bell System Technical Journal*, v. 58, n. 5, pp. 963–993, May 1979.

[16] REBHI, S., BARRAK, R., MENIF, M. "LMS-based Digital Pre-equalizer for Cognitive RoF System". In: *18th International Conference on Transparent Optical Networks*, pp. 1–4, Trento, Italy, July 2016.

[17] SHAH, S. M. "Riemann–Liouville Operator-based Fractional Normalised Least Mean Square Algorithm with Application to Decision Feedback Equalisation of Multipath Channels", *IET Signal Processing*, v. 10, n. 6, pp. 575–582, July 2016.

[18] GARRETT, D., WOODWARD, G., DAVIS, L., et al. "A 28.8 Mb/s 4/spl times/4 MIMO 3G High-speed Downlink Packet Access Receiver with Normalized Least Mean Square Equalization". In: *2004 IEEE International Solid-State Circuits Conference*, pp. 420–536, San Francisco, USA, February 2004.

[19] RUPP, M., HAUSBERG, F. "LMS Algorithmic Variants in Active Noise and Vibration Control". In: *22th European Signal Processing Conference*, pp. 691–695, Lisbon, Portugal, September 2014.

[20] KUO, S. M., MORGAN, D. R. *Active Noise Control Systems: Algorithms and DSP Implementations*. 1st ed. New York, USA, Wiley, 1996.

[21] LI TAN, JIANG, J. "Adaptive Volterra Filters for Active Control of Nonlinear Noise Processes", *IEEE Transactions on Signal Processing*, v. 49, n. 8, pp. 1667–1676, August 2001.

[22] BISWAS, U., DAS, A., DEBNATH, S., et al. "ECG Signal Denoising by Using Least-mean-square and Normalised-least-mean-square Algorithm based Adaptive Filter". In: *2014 International Conference on Informatics, Electronics & Vision*, pp. 1–6, Dhaka, Bangladesh, July 2014.

[23] WESTWICK, D. T., MAUNDY, B., SALMEH, R. "Approximate LMS Tuning of Continuous Time Filters: Convergence and Sensitivity Analysis",

IEE Proceedings - Circuits, Devices and Systems, v. 152, n. 1, pp. 1–6, February 2005.

[24] CIOCHINĂ, S., PALEOLOGU, C., BENESTY, J., et al. "A Family of Optimized LMS-based Algorithms for System Identification". In: *24th European Signal Processing Conference*, pp. 1803–1807, Budapest, Hungary, September 2016.

[25] WIDROW, B., MCCOOL, J., LARIMORE, M. G., et al. "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter", *Proceedings of the IEEE*, v. 64, n. 8, pp. 1151–1162, August 1976.

[26] MOHAMMAD, T. I., ZAINOL, A. A. R. "MI-NLMS Adaptive Beamforming Algorithm for Smart Antenna System Applications", *Journal of Zhejiang University-SCIENCE A*, v. 7, n. 10, pp. 1709–1716, October 2006.

[27] DUTTWEILER, D. L. "Proportionate Normalized Least-mean-squares Adaptation in Echo Cancelers", *IEEE Transactions on Speech and Audio Processing*, v. 8, n. 5, pp. 508–518, September 2000.

[28] LIMA, M. V. S., FERREIRA, T. N., MARTINS, W. A., et al. "Performance Evaluation of Adaptive Filters for Sparse Wireless Channel Estimation". In: *25th European Signal Processing Conference*, pp. 2601–2065, Kos, Greece, August 2017.

[29] WUNDER, G., BOCHE, H., STROHMER, T., et al. "Sparse Signal Processing Concepts for Efficient 5G System Design", *IEEE Access*, v. 3, pp. 195–208, February 2015.

[30] SUBRAHMANYA, N., SHIN, Y. C. "Sparse Multiple Kernel Learning for Signal Processing Applications", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 32, n. 5, pp. 788–798, May 2010.

[31] COTTER, S. F., RAO, B. D. "Sparse Channel Estimation via Matching Pursuit with Application to Equalization", *IEEE Transactions on Communications*, v. 50, n. 3, pp. 374–377, August 2002.

[32] NOSE-FILHO, K., TAKAHATA, A. K., LOPES, R., et al. "Improving Sparse Multichannel Blind Deconvolution with Correlated Seismic Data: Foundations and Further Results", *IEEE Signal Processing Magazine*, v. 35, n. 2, pp. 41–50, March 2018.

[33] YAZDANPANAH, H., CARINI, A., LIMA, M. V. S. "l_0-Norm Adaptive Volterra Filters". In: *27th European Signal Processing Conference*, pp. 1–5, A Coruña, Spain, September 2019.

[34] ELAD, M. *Sparse and Redundant Representations: from Theory to Applications in Signal and Image Processing*. 1st ed. New York, USA, Springer Science & Business Media, 2010.

[35] ELDAR, Y. C., KUTYNIOK, G. *Compressed Sensing: Theory and Applications*. 1st ed. Cambridge, UK, Cambridge University Press, 2012.

[36] HUANG, K., AVIYENTE, S. "Sparse Representation for Signal Classification". In: *Advances in Neural Information Processing Systems*, pp. 609–616, Vancouver, Canada, December 2006.

[37] BARANIUK, R. G., CANDES, E., ELAD, M., et al. "Applications of Sparse Representation and Compressive Sensing [Scanning the Issue]", *Proceedings of the IEEE*, v. 98, n. 6, pp. 906–909, May 2010.

[38] GAY, S. "An Efficient, Fast Converging Adaptive Filter for Network Echo Cancellation". In: *Thirty-Second Asilomar Conference on Signals, Systems amp, Computers*, v. 1, pp. 394–398, Pacific Grove, USA, November 1998.

[39] BENESTY, J., GAY, S. L. "An Improved PNLMS algorithm". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, v. 2, pp. 1881–1884, Dallas, USA, May 2002.

[40] DENG, H., DOROSLOVACKI, M. "Improving Convergence of the PNLMS Algorithm for Sparse Impulse Response Identification", *IEEE Signal Processing Letters*, v. 12, n. 3, pp. 181–184, March 2005.

[41] LIU, L., FUKUMOTO, M., SAIKI, S. "An Improved mu-law Proportionate NLMS Algorithm". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3797–3800, Las Vegas, USA, March 2008.

[42] PALEOLOGU, C., CIOCHINA, S., BENESTY, J. "An Efficient Proportionate Affine Projection Algorithm for Echo Cancellation", *IEEE Signal Processing Letters*, v. 17, n. 2, pp. 165–168, February 2010.

[43] SAKHNOV, K. "An Improved Proportionate Affine Projection Algorithm for Network Echo Cancellation". In: *2008 15th International Conference on Systems, Signals and Image Processing*, pp. 125–128, Bratislava, Slovakia, June 2008.

[44] DE SOUZA, F. D. C., SEARA, R., MORGAN, D. R. "An Enhanced IAF-PNLMS Adaptive Algorithm for Sparse Impulse Response Identification", *IEEE Transactions on Signal Processing*, v. 60, n. 6, pp. 3301–3307, June 2012.

[45] GANSLER, T., BENESTY, J., GAY, S. L., et al. "A Robust Proportionate Affine Projection Algorithm for Network Echo Cancellation". In: *2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, v. 2, pp. II793–II796, Istanbul, Turkey, June 2000. IEEE.

[46] DE SOUZA, F. D. C., TOBIAS, O. J., SEARA, R., et al. "A PNLMS Algorithm with Individual Activation Factors", *IEEE Transactions on Signal Processing*, v. 58, n. 4, pp. 2036–2047, December 2009.

[47] PETRAGLIA, M., HADDAD, D. "New Adaptive Algorithms for Identification of Sparse Impulse Responses – Analysis and Comparisons". In: *7th International Symposium on Wireless Communication Systems*, pp. 384–388, York, UK, September 2010.

[48] BOYD, S. *Convex Optimization*. 1st ed. Cambridge, UK, Cambridge University Press, 2004.

[49] DONOHO, D. L., HUO, X. "Uncertainty Principles and Ideal Atomic Decomposition", *IEEE Transactions on Information Theory*, v. 47, n. 7, pp. 2845–2862, November 2001.

[50] NOSE-FILHO, K., JUTTEN, C., ROMANO, J. M. T. "Sparse Blind Deconvolution Based on Scale Invariant Smoothed l_0-norm". In: *22nd European Signal Processing Conference*, pp. 461–465, September 2014.

[51] LIMA, M. V. S., FERREIRA, T. N., MARTINS, W. A., et al. "Sparsity-aware Data-selective Adaptive Filters", *IEEE Transactions on Signal Processing*, v. 62, n. 17, pp. 4557–4572, September 2014.

[52] NOSE-FILHO, K., ROMANO, J. M. T. "On l_p-norm Sparse Blind Deconvolution". In: *IEEE International Workshop on Machine Learning for Signal Processing*, pp. 1–6, September 2014.

[53] GU, Y., JIN, J., MEI, S. "l_0 norm Constraint LMS Algorithm for Sparse System Identification", *IEEE Signal Processing Letters*, v. 16, n. 9, pp. 774–777, September 2009.

[54] CHEN, Y., GU, Y., HERO, A. "Sparse LMS for System Identification". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3125–3128, Taipei, Taiwan, April 2009.

[55] RUPP, M., SCHWARZ, S. "A Tensor LMS Algorithm". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3347–3351, Brisbane, Australia, August 2015. IEEE.

[56] YAZDANPANAH, H., DINIZ, P. S. R., LIMA, M. V. S. "A Simple Set-Membership Affine Projection Algorithm for Sparse System Modeling". In: *24th European Signal Processing Conference (EUSIPCO 2016)*, pp. 1798–1802, Budapest, Hungary, September 2016.

[57] YAZDANPANAH, H., APOLINARIO JR., J. A., DINIZ, P. S. R., et al. "l0-norm Feature LMS Algorithms". In: *2018 IEEE Global Conference on Signal and Information Processing*, pp. 311–315, Anaheim, USA, November 2018.

[58] LIU, W., PRINCIPE, J. C., HAYKIN, S. *Kernel Adaptive Filtering: A Comprehensive Introduction*. Adaptive and Cognitive Dynamic Systems: Signal Processing, Learning, Communications and Control. 1st ed. New York, USA, Wiley, 2010.

[59] HOSHUYAMA, O., GOUBRAN, R., SUGIYAMA, A. "A Generalized Proportionate Variable Step-size Algorithm for Fast Changing Acoustic Environments". In: *IEEE International Conference on Acoustics, Speech, and Signal Processing*, v. 4, pp. 161–164, Montreal, Canada, May 2004.

[60] KHONG, A. W. H., NAYLOR, P. A. "Efficient Use of Sparse Adaptive Filters". In: *Fortieth Asilomar Conference on Signals, Systems and Computers*, pp. 1375–1379, Pacific Grove, USA, November 2006.

[61] BARANIUK, R. G., CANDES, E., ELAD, M., et al. "Applications of Sparse Representation and Compressive Sensing [Scanning the Issue]", *Proceedings of the IEEE*, v. 98, n. 6, pp. 906–909, June 2010.

[62] OLINTO, K. D. S., HADDAD, D. B., PETRAGLIA, M. R. "Transient Analysis of l0-LMS and l0-NLMS Algorithms", *Signal Processing*, v. 127, pp. 217–226, October 2016.

[63] STEWART, J. *Calculus*. 7th ed. Boston, USA, Cengage Learning, 2019.

[64] TRZASKO, J., MANDUCA, A. "Highly Undersampled Magnetic Resonance Image Reconstruction via Homotopic l_0-minimization", *IEEE Transactions on Medical Imaging*, v. 28, n. 1, pp. 106–121, January 2009.

[65] RAN MENG, DE LAMARE, R. C., NASCIMENTO, V. H. "Sparsity-aware Affine Projection Adaptive Algorithms for System Identification". In: *Sensor Signal Processing for Defence*, pp. 1–5, London, UK, September 2011.

[66] ABRAMOWITZ, M., STEGUN, I. A. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables*. 1st ed. New York, USA, Dover Publications, 1965.

[67] ARFKEN, G. B. *Mathematical Methods for Physicists*. 5th ed. Cambridge, USA, Academic Press, 2000.

[68] DIRAC, P. A. M. *The Principles of Quantum Mechanics (International Series of Monographs on Physics)*. 4th ed. Oxford, UK, Clarendon Press, 1982.

[69] CHEN, Y., GU, Y., HERO, A. O. "Regularized Least-Mean-Square Algorithms", *arXiv e-prints*, pp. 1–9, December 2010.

[70] KOPSINIS, Y., SLAVAKIS, K., THEODORIDIS, S. "Online Sparse System Identification and Signal Reconstruction Using Projections onto Weighted l_1 Balls", *IEEE Transactions on Signal Processing*, v. 59, n. 3, pp. 936–952, March 2011.

[71] DE LAMARE, R., SAMPAIO-NETO, R. "Sparsity-aware Adaptive Algorithms Based on Alternating Optimization and Shrinkage", *IEEE Signal Processing Letters*, v. 21, n. 2, pp. 225–229, February 2014.

[72] LIMA, M. V. S., MARTINS, W. A., DINIZ, P. S. R. "Affine Projection Algorithms for Sparse System Identification". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5666–5670, Vancouver, Canada, May 2013.

[73] LIMA, M. V. S., SOBRON, I., MARTINS, W. A., et al. "Stability and MSE Analyses of Affine Projection Algorithms for Sparse System Identification". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6399–6403, Florence, Italy, May 2014.

[74] CANDES, E. J., WAKIN, M. B., BOYD, S. P. "Enhancing Sparsity by Reweighted l_1 Minimization", *Journal of Fourier Analysis and Applications*, v. 14, n. 5, pp. 877–905, December 2008.

[75] GASSO, G., RAKOTOMAMONJY, A., CANU, S. "Recovering Sparse Signals with a Certain Family of Nonconvex Penalties and DC Programming", *IEEE Transactions on Signal Processing*, v. 57, n. 12, pp. 4686–4698, December 2009.

[76] YAZDANPANAH, H., DINIZ, P. S. R. "Recursive Least-squares Algorithms for Sparse System Modeling". In: *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3879–3883, New Orleans, USA, March 2017.

[77] MENG, R., DE LAMARE, R., NASCIMENTO, V. "Sparsity-aware Affine Projection Adaptive Algorithms for System Identification". In: *Sensor Signal Processing for Defence*, pp. 1–5, London, U.K., September 2011.