Pontifícia Universidade Católica
do Rio de Janeiro

André Robert Flores Manrique

Data-Selective Linear and Kernel-Based Adaptive Algorithms

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Engenharia Elétrica of the Departamento de Engenharia Elétrica, PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica.

Advisor: Prof. Rodrigo Caiado de Lamare

Rio de Janeiro
March 2017

Pontifícia Universidade Católica
do Rio de Janeiro

André Robert Flores Manrique

Data-Selective Linear and Kernel-Based Adaptive Algorithms

Dissertation presented to the Programa de Pós–graduação em Engenharia Elétrica of the Departamento de Engenharia Elétrica, PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Engenharia Elétrica. Approved by the undersigned Examination Committee.

Prof. Rodrigo Caiado de Lamare
Orientador
Centro de Estudos em Telecomunicações – PUC-Rio

Prof. Wouter Caarls
Departamento de Engenheria Elétrica – PUC-Rio

Dr. Lukas Landau
Centro de Estudos em Telecomunicações – PUC-Rio

Prof. Jose Antonio Apolinário Jr.
Instituto Militar de Engenharia – IME

Prof. Josef Anton Nossek
Universidade Federal do Ceara – UFC

Prof. Márcio da Silveira Carvalho
Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, March 24th, 2017

André Robert Flores Manrique

The author graduated in Telecommunication Engineering from the San Pablo University, in Arequipa, Peru, 2014

Ficha Catalográfica

Flores Manrique, André Robert

Data-Selective Linear and Kernel-Based Adaptive Algorithms / André Robert Flores Manrique; advisor: Rodrigo Caiado de Lamare. – 2017.

v., 101 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Elétrica.

Inclui bibliografia

1. Engenharia Elétrica – Teses. 2. Algoritmos com seleção de dados;. 3. Métodos kernel;. 4. Esparsidade;. 5. Algoritmos que exploram a esparsidade;. 6. Processamento adaptativo de sinais;. I. de Lamare, Rodrigo C.. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Elétrica. III. Título.

CDD: 621.3

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisor Prof. Rodrigo C. de Lamare not only for the continuous support, but also for his motivation, enthusiasm and knowledge, without which much of this work would not have been possible.

Further thanks to all the professors that have contributed to my academic and professional career, especially to the professors of PUC-Rio.

My sincere thanks to the CNPq, FAPERJ, and PUC-Rio, for the financial support.

I also want to say thanks to my colleagues and friends of CETUC for their support and help.

Last but not least, I want to express my profound gratitude to my family, especially my parents and my brother, for their encouragement and for always being there for me.

Abstract

Flores Manrique, André Robert; de Lamare, Rodrigo C. (Advisor). **Data-Selective Linear and Kernel-Based Adaptive Algorithms**. Rio de Janeiro, 2017. 101p. MSc. Dissertation – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

In this dissertation, several data-selective adaptive signal processing algorithms are derived and investigated for solving two different problems. The first one involves scenarios handling sparse systems, where we introduce a framework in which a general penalty function is incorporated into the cost function for exploiting the sparsity of the model. Under this scope, we propose three algorithms with an adjustable penalty function, the first one based on the $l_1 - norm$, which we term zero-attracting SM-NLMS with adjustable penalty function (ZA-SM-NLMS-ADP). The second algorithm is based on the log-sum penalty function and the third one on the $l_0 - norm$, named reweighted ZA-SM-NLMS (RZA-SM-NLMS-ADP) and the exponential ZA-SM-NLMS (EZA-SM-NLMS-ADP), respectively. We also carry out a statistical analysis of the sparsity-aware SM-NLMS algorithms with a general penalty function, arriving at mathematical expressions for the mean-square error at steady state. The second problem addressed considers nonlinear adaptive algorithms based on kernel functions. In this context, we develop two data selective algorithms, the Set-Membership Normalized Kernel Least Mean Squares (SM-NKLMS) algorithm and the Set-Membership Kernel Affine Projection (SM-KAP) algorithm, which have the capability of naturally limiting the growing structure created by the kernels, dealing with one of the major problems presented when working with kernel algorithms. The kernel algorithms developed have been tested for a time series prediction task. A statistical analysis of the proposed SM-NKLMS algorithm is also developed. Simulation results show that the proposed algorithms, outperform standard linear and nonlinear adaptive algorithms in both convergence rate and steady state performance.

Keywords

Set-membership algorithms; Kernel methods; sparsification; Sparsity-aware algorithms; Adaptive signal processing;

Resumo

Flores Manrique, André Robert; de Lamare, Rodrigo C.. **Data-Selective Linear and Kernel-Based Adaptive Algorithms**. Rio de Janeiro, 2017. 101p. Dissertação de Mestrado – Departamento de Engenharia Elétrica, Pontifícia Universidade Católica do Rio de Janeiro.

Nesta dissertação, diversos algoritmos adaptativos para processamento de sinais com seleção de dados são desenvolvidos e estudados, com o objetivo de resolver dois problemas diferentes. O primeiro problema envolve ambientes com sistemas esparsos, onde uma função penalidade é incorporada na função de custo para aproveitar a esparsidade do modelo. Nesta perspectiva, são propostos três algoritmos com função penalidade ajustável, o primeiro baseado na função penalidade l_1 é denominado SM-NLMS com atração para zero e função penalidade ajustável (ZA-SM-NLMS-ADP). O segundo algoritmo está baseado na função penalidade log-sum e o terceiro na função penalidade l_0 , denominados SM-NLMS com atração ponderada para zero e função de penalidade ajustável (RZA-SM-NLMS-ADP) e SM-NLMS com atração para zero exponencial e função de penalidade ajustável (EZA-SM-NLMS-ADP), respectivamente. Além disso, foi desenvolvida uma análise estatística do algoritmo SM-NLMS com uma função penalidade genérica, obtendo expressões matemáticas para o erro médio quadrático em estado estacionário. O segundo problema abordado, considera algoritmos adaptativos não lineares baseados em funções de kernels. Neste contexto, são desenvolvidos dois algoritmos com seleção de dados, o algoritmo SM-NKLMS e o algoritmo SM-KAP, os quais possuem a capacidade de limitar o crescimento da estrutura criada pelas funções de kernels, tratando um dos maiores problemas que surge quando se utilizam algoritmos baseados em kernels. Os algoritmos baseados em kernels foram testados para predição de séries temporais. Também é realizada uma análise estatística do algoritmo SM-NKLMS. As simulações mostram que os algoritmos desenvolvidos superam os algoritmos lineares e não lineares convencionais tanto na velocidade de convergência quanto no erro médio quadrático atingido.

Palavras-chave

Algoritmos com seleção de dados; Métodos kernel; Esparsidade; Algoritmos que exploram a esparsidade; Processamento adaptativo de sinais;

Summary

Figure list

Table list

List of Abreviations

ADP – Adjustable Penalty Function
ALD – Approximate Linear Dependency
APA – Affine Projection Algorithm
CC – Coherence Criterion
EMSE – Excess of Mean Square Error
EZA–Exponential Zero-Attracting
IMPNLMS – Improved Mu-Law Proportionate Normalized Least-Mean-Squares
KAF – Kernel Adaptive Filtering
KAPA – Kernel Affine Projection Algorithm
KLMS – Kernel Least-Mean-Squares
LMS – Least-Mean-Square
MPNLMS – Mu-Law Proportionate Normalized Least-Mean-Square
MMSE–Minimum Mean Square Error
MSD – Mean Square Deviation
MSE – Mean Square Error
NC – Novelty Criterion
NLMS – Normalized Least-Mean-Square
PAP – Proportionate Affine Projection
PNLMS – Proportionate Normalized Least-Mean-Square
RZA–Reweighted Zero-Attracting
SC – Surprise Criterion
SM-APA – Set-Membership Affine Projection Algorithm
SM-KAP –Set-Membership Kernel Affine Projection
SM-NKLMS – Set-Membership Normalized Kernel Least-Mean-Squares
SM-NLMS – Set-Membership Normalized Least-Mean-Square
SM-PAP – Set-Membership Proportionate Affine Projection
SM-PNLMS – Set-Membership Proportionate Normalized Least-Mean-Square
ZA–Zero-Attracting

1
Introduction

Over the last forty years, the field of digital signal processing has developed in a very fast way. Due to the incremental growth of knowledge in this area, some specialized topics have turned into fields themselves. One example of this is adaptive signal processing. The designer can usually choose the most appropriate algorithm to process a signal whose statistical properties are fixed and known (e.g. the Wiener filter which minimizes the mean square error between the output of the filter and some desired signal). However, if these properties change over time or if it is not possible to obtain a sufficiently accurate estimate of these values, then the fixed algorithms used can not process the signals efficiently. The solution to overcome this kind of problems is to use adaptive algorithms which are capable of changing or updating their characteristics automatically. In this sense, adaptive algorithms are required whether the fixed specifications are unknown or in time-variant scenarios.

In general, adaptive algorithms perform two basic steps. The first one involves the computation of the output signal, generated from the input signal. The second step is an adaptation process that adjust the parameters of the system to minimize a desired cost function. The ability of an adaptive algorithm to operate satisfactorily in an unknown environment and track time variations of statistics makes the adaptive algorithms a powerful tool for signal processing and control applications. As a consequence, adaptive algorithms were successfully implemented in a wide variety of devices for different applications fields such as communications, radar and biomedical engineering.

1.1
Motivation

Conventional adaptive algorithms perform an update whenever new data arrive. Each executed update demands computational resources. However, the computational load is a critical factor, when developing a system. Because of this fact, we need to implement a strategy to manage the computational resources. A useful strategy to obtain some control on the computational resources is to develop algorithms that implement sparse updates. This results

in a more efficient management of the computational resources. A processor could perform multiple tasks, taking into account that the updates are sparse. The idea of sparse updates based on a bounded noise was first reported in [1]. Later the set-membership NLMS algorithm was formally described in [2] and was used since then in many applications [3–7] due its good performance as compared to conventional algorithms. The sparse update scheme promoted by this kind of algorithms reduces the computational load, resulting in faster algorithms.

Set-membership adaptive algorithms were also used for sparse system identification, where only a few large coefficients are interspersed among many negligible ones. However, the parameter that controls the strength of the penalty function is fixed and setting this parameter is a critical step to obtain a good performance of the algorithm. In this sense, an automatic adjustment of the parameters of the penalty function will enhance the overall performance of the algorithm. Also, to the best of our knowledge, no statistical analysis was carried out for sparsity-aware set-membership adaptive algorithms. The main difficulty of this analysis lies in the nonlinearity of the update rule and in the sparse update scheme implemented.

Although linear adaptive systems are a good alternative to solve many kinds of problems, there exist several applications where nonlinear techniques show advantages. For example, in communications if the signal to noise ratio is not high enough, the use of linear adaptive equalizers results in a poor performance in terms of bit error rate [8]. Also, for a time series prediction task, a nonlinear structure can show better performance. Motivated by this fact, in the last ten years, kernel adaptive filtering has been the focus of many researchers and many algorithms were proposed [9–12]. However, the computational complexity of kernel adaptive algorithms is significantly higher than their linear counterparts. In this context, a sparse update scheme will reduce the computational load of these algorithms. In this dissertation we propose kernel adaptive algorithms with sparse updates for reducing the computational complexity.

1.2 Contributions

The contributions of this dissertation can be summarized as follows:

– A general framework for data-selective algorithms, that exploits the sparsity of the model has been derived following a gradient descent approach. The development led us to a modified expression for the step-size and also to an adjustable penalty function. Then we employed three

different penalty functions over the established framework to propose the ZA-SM-NLMS-ADP, the RZA-SM-NLMS-ADP and the EZA-SM-NLMS-ADP algorithms. The proposed algorithms have been used in a system identification application and presented a faster convergence rate than conventional algorithms.

– A statistical analysis for data-selective algorithms with a general penalty function is carried out. A mathematical model for the mean square error (MSE) at steady state for an algorithm with a general penalty function was presented. The ZA-SM-NLMS algorithm has been chosen to test the proposed model. Simulations show agreement with the mathematical formulation.

– Two data-selective algorithms based on kernel functions are proposed, namely the SM-NKLMS and the SM-KAP. The main advantage of the developed algorithms is that the sparse update scheme limit the growing structure generated by the kernel expansion. A statistical analysis for the SM-NKLMS algorithm is also carried out. Simulation results show that the proposed algorithms outperform the standard kernel adaptive algorithms in both convergence rate and steady state performance.

1.3 Dissertation Outline

This dissertation is organized as follows:

– In Chapter 2, a review of adaptive signal processing is given. The conventional sparsity-aware algorithms are introduced, as well as the conventional kernel adaptive algorithms.

– In Chapter 3, the proposed data-selective sparsity-aware algorithms are described. The developed framework considers the adjustment of the parameters of the penalty function and the step-size to achieve a better performance. A statistical analysis is carried out, resulting in a mathematical model for the MSE performance at steady state.

– In Chapter 4, two data-selective kernel adaptive algorithms are presented. Simulations involving a time series prediction task are carried out to compare the performance of the proposed algorithms against the existing kernel algorithms. A statistical analysis is also carried out.

– In Chapter 5, conclusions of this work are presented and future directions for this research topic are discussed.

1.4 List of Publications

Some of the results in this dissertation have been published, or will be submitted to publications.

Conference Papers:

- A. Flores and R.C. de Lamare, "Set-Membership Kernel Adaptive algorithms," IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), New Orleans, USA, 2017
- A. Flores and R.C. de Lamare, "Sparsity-Aware Set-Membership Adaptive Algorithms with Adjustable Penalties", submitted to the International Conference on Digital Signal Processing, 2017.

Journal Papers:

- A. Flores and R.C. de Lamare, "Sparsity-aware set-membership adaptive algorithms with adjustable penalties: design and analysis" (Under Preparation).
- A. Flores and R.C. de Lamare, "Data-selective kernel adaptive algorithms: design and analysis" (Under Preparation).

1.5 Notation

$\mathbf{a}$	a vector (boldface lower case letters).
$\mathbf{A}$	a matrix (boldface uppercase letters).
$\mathbf{I}_N$	$N \times N$ identity matrix.
$(\cdot)^{\mathrm{T}}$	Matrix transpose.
$\mathbb{E}[\cdot]$	Expectation operator.
$\|\cdot\|$	$l_2 - norm$

2
Adaptive Signal Processing Review

In this chapter, a review of adaptive signal processing techniques is given. First we revisit fundamental concepts involving machine learning. Then we examine linear adaptive algorithms, where we also discuss the two principal techniques used to exploit the sparsity of a system. Finally, we extend our discussion to the nonlinear case of adaptive signal processing, focusing on Kernel-based algorithms.

2.1
Machine Learning

Learning is the activity of acquiring new or modifying the existing knowledge. With the improvements of technology, the ability to learn is not only limited to human beings and animals, but also include some artificial machines. Thanks to the learning process, a system is able to adapt the parameters of a model according to the environment. The best way to deal with scenarios that are constantly changing is to adapt the system parameters through learning [8, 13]. There are three ways to perform learning:

- Supervised learning: requires a set of desired responses, which are going to lead the learning process.
- Unsupervised learning: a self-organized process that does not require a desired response.
- Reinforcement learning: when there is some feedback that classifies the responses as good ones or bad ones, but there is no explicit input-output scheme. It is a scheme based on rewards.

All the algorithms presented in this work use supervised learning where the knowledge is represented by a set of training samples. Another characteristic of the methods used is that they perform sequential learning. This means that the training samples arrive consecutively over time. After receiving one sample of data the learning algorithm gives an estimate of the output. Then the estimated value is compared to the true value, resulting in an update of the current parameters. The set of samples described before is formed by the data pairs

$$\{\mathbf{x}(1), d(1)\}, \{\mathbf{x}(2), d(2)\}, \cdots, \{\mathbf{x}(i), d(i)\},$$

where $\mathbf{x}(i)$ represents the input vector and $d(i)$ is the desired response for that input. A tapped-delay-line is used to form the input vector. This sequence is going to produce different sets of parameters

$$\mathbf{w}(1), \mathbf{w}(2), \cdots, \mathbf{w}(i),$$

where $\mathbf{w}(i)$ not only depends on the pair $\{\mathbf{x}(i), d(i)\}$, but also on the previous parameters $\mathbf{w}(i-1)$. It is possible to extend the approach introduced so that $\mathbf{w}(i)$ will depend on a set of training samples pairs instead of just one sample pair.

2.2 Adaptive Signal Processing Fundamentals

The general structure of an adaptive filter is shown in Figure 2.1. The structure embodies a set of adjustable parameters, also known as weights, denoted by vector $\mathbf{w}(i)$. The iteration or time instant is defined by the variable i. The input signal and the adaptive filtering output are represented by $\mathbf{x}(i)$ and $y(i)$, respectively. The desired signal is denoted by $d(i)$.

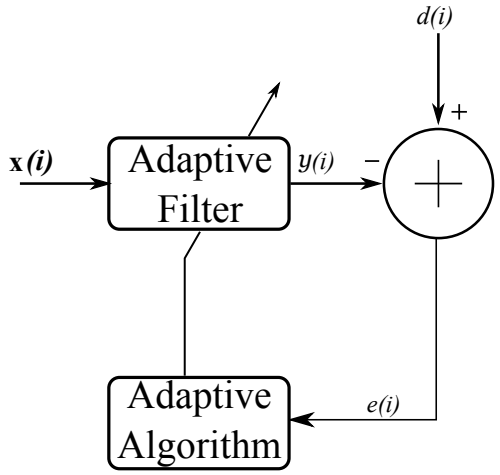

Figure 2.1: Adaptive Filtering Scheme.

A linear combiner computes the output of the system as follows:

$$y(i) = \mathbf{w}^{\mathrm{T}}(i)\,\mathbf{x}(i). \tag{2-1}$$

Given the output of the system, the error signal can be calculated by the following equation:

$$e(i) = d(i) - y(i), \tag{2-2}$$

used to establish an objective function that defines the updating of the filter coefficients. In general, the adaptive algorithm updates the filter coefficients in such a way that the objective function is minimized. Minimizing the objective function requires a search for the best matching between the adaptive algorithm output and the desired signal [8]. For the algorithms in this work, we are going to use the mean square error as the objective function:

$$J(i) = \mathbb{E}\left[e^2\left(i\right)\right]. \tag{2-3}$$

The ensemble average of the square error, given by equation (2-3), plotted versus the time i, traces the learning curve of the adaptive process.

2.2.1
The Least-Mean-Square (LMS) Algorithm

The LMS algorithm is the simplest form of an adaptive algorithm [14]. This algorithm minimize the instantaneous objective function

$$J\left(i\right) = \frac{1}{2}e^2\left(i\right). \tag{2-4}$$

Taking the gradient of the objective function with respect to the weights, we obtain

$$\frac{\partial}{\partial \mathbf{w}(i)} J\left(i\right) = -e(i)\mathbf{x}(i). \tag{2-5}$$

The gradient updates the weights through the following recursion:

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \mu e(i)\mathbf{x}(i), \tag{2-6}$$

where μ is the step-size that controls the convergence of the LMS algorithm.

2.2.2
The Normalized Least-Mean-Square (NLMS) Algorithm

The NLMS algorithm, proposed in [15], can be derived by solving the following constrained optimization:

$$\begin{gathered}\min_{\mathbf{w}(i+1)} \parallel \mathbf{w}\left(i+1\right) - \mathbf{w}\left(i\right) \parallel^2 \\ \text{subject to} \\ d\left(i\right) - \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right) = 0.\end{gathered} \tag{2-7}$$

The constrained optimization above can be transformed into an unconstrained optimization using the method of Lagrange multipliers which considers the Lagrangian given by

$$\mathcal{L}\left(\mathbf{w}\left(i+1\right),\lambda\right) = \parallel \mathbf{w}\left(i+1\right) - \mathbf{w}\left(i\right) \parallel^2 + \lambda\left(d\left(i\right) - \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right)\right). \quad (2\text{-}8)$$

Taking the gradient from the last equation with respect to $\mathbf{w}\left(i+1\right)$ and λ, we get

$$\frac{\partial\mathcal{L}\left(\mathbf{w}\left(i+1\right),\lambda\right)}{\partial\mathbf{w}\left(i+1\right)} = 2\left(\mathbf{w}\left(i+1\right) - \mathbf{w}\left(i\right)\right) - \lambda\mathbf{x}\left(i\right), \quad (2\text{-}9)$$

$$\frac{\partial\mathcal{L}\left(\mathbf{w}\left(i\right),\lambda\right)}{\partial\lambda} = d\left(i\right) - \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right). \quad (2\text{-}10)$$

Equating the resulting terms to zero leads to:

$$\mathbf{w}\left(i+1\right) = \mathbf{w}\left(i\right) + \frac{\lambda}{2}\mathbf{x}\left(i\right), \quad (2\text{-}11)$$

$$d\left(i\right) = \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right). \quad (2\text{-}12)$$

By premultiplying equation (2-11) by $\mathbf{x}^{\mathrm{T}}\left(i\right)$ we obtain

$$d\left(i\right) = \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i\right) + \frac{\lambda}{2}\mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{x}\left(i\right), \quad (2\text{-}13)$$

$$\frac{\lambda}{2} = \frac{e\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2}. \quad (2\text{-}14)$$

Substituting the last equation in (2-11) leads to the NLMS update equation given by

$$\mathbf{w}\left(i+1\right) = \mathbf{w}\left(i\right) + \mu\frac{e\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2}\mathbf{x}\left(i\right), \quad (2\text{-}15)$$

where μ is the step-size incorporated for controlling the convergence of the algorithm.

2.2.3
The Affine Projection Algorithm

The affine projection (AP) algorithm [16] is based on data reuse to accelerate the learning process. The main difference between the AP algorithm and the LMS algorithm is that the AP algorithm reuses multiple samples to achieve a faster convergence speed rather than simply use the input data of the current time instant. Let us consider a matrix $\mathbf{X}\left(i\right)$ containing the last K input vectors as described by,

$$\mathbf{X}\left(i\right) = \begin{bmatrix} x\left(i\right) & x\left(i-1\right) & \cdots & x\left(i-K+1\right) \\ x\left(i-1\right) & x\left(i-2\right) & \cdots & x\left(i-K\right) \\ \vdots & \vdots & \ddots & \vdots \\ x\left(i-N+1\right) & x\left(i-N\right) & \cdots & x\left(i-K-N+1\right) \end{bmatrix}, \quad (2\text{-}16)$$

where N represents the length of a single input vector. A compact expression of matrix $\mathbf{X}$ can be formulated as follows:

$$\mathbf{X}(i) = \left[\begin{array}{cccc} \mathbf{x}(i) & \mathbf{x}(i-1) & \cdots & \mathbf{x}(i-K+1) \end{array} \right]. \tag{2-17}$$

Let us now define the output of the system as described by

$$\boldsymbol{y}(i) = \mathbf{X}^{\mathrm{T}}(i)\,\mathbf{w}(i) = \begin{bmatrix} y_0(i) \\ y_1(i) \\ \vdots \\ y_{K-1}(i) \end{bmatrix} \tag{2-18}$$

and the desired and the error signal, respectively,

$$\boldsymbol{d}(i) = \begin{bmatrix} d(i) \\ d(i-1) \\ \vdots \\ d(i-K+1) \end{bmatrix}, \tag{2-19}$$

$$\boldsymbol{e}(i) = \boldsymbol{d}(i) - \boldsymbol{y}(i) = \begin{bmatrix} e_0(i) \\ e_1(i) \\ \vdots \\ e_{K-1}(i) \end{bmatrix}. \tag{2-20}$$

As stated in [8], the goal of the algorithm is to solve the optimization problem

$$\begin{gathered} \min_{\mathbf{w}(i+1)} \ \frac{1}{2} \parallel \mathbf{w}(i+1) - \mathbf{w}(i) \parallel^2 \\ \text{subject to} \\ \boldsymbol{d}(i) - \mathbf{X}^{\mathrm{T}}(i)\,\mathbf{w}(i+1) = \mathbf{0}. \end{gathered} \tag{2-21}$$

Using the method of Lagrange multipliers to solve equation (2-21), we obtain the Lagrangian function given by

$$\mathcal{L}(\mathbf{w}(i+1)) = \frac{1}{2} \parallel \mathbf{w}(i+1) - \mathbf{w}(i) \parallel^2 + \boldsymbol{\lambda}^{\mathrm{T}}(i) \Big(\boldsymbol{d}(i) - \mathbf{X}^{\mathrm{T}}(i)\,\mathbf{w}(i+1) \Big). \tag{2-22}$$

Computing the gradient of $\mathcal{L}(\mathbf{w}(i+1))$ with respect to $\mathbf{w}(i+1)$ and equating it to zero we arrive at:

$$\frac{\partial \mathcal{L}(\mathbf{w}(i+1))}{\partial \mathbf{w}(i+1)} = \frac{1}{2}(2\mathbf{w}(i+1) - 2\mathbf{w}(i)) - \mathbf{X}(i)\,\boldsymbol{\lambda}(i) = \mathbf{0}, \tag{2-23}$$

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \mathbf{X}(i)\,\boldsymbol{\lambda}(i). \tag{2-24}$$

Substituting equation (2-24) in (2-21), we get

$$\boldsymbol{d}(i) - \mathbf{X}^{\mathrm{T}}(i)\left(\mathbf{w}(i) + \mathbf{X}(i)\,\boldsymbol{\lambda}(i)\right) = \mathbf{0}, \tag{2-25}$$

$$\boldsymbol{d}(i) - \mathbf{X}^{\mathrm{T}}(i)\,\mathbf{w}(i) = \mathbf{X}^{\mathrm{T}}(i)\,\mathbf{X}(i)\,\boldsymbol{\lambda}(i), \tag{2-26}$$

$$\boldsymbol{e}(i) = \mathbf{X}^{\mathrm{T}}(i)\,\mathbf{X}(i)\,\boldsymbol{\lambda}(i). \tag{2-27}$$

Solving for $\boldsymbol{\lambda}(i)$, we obtain

$$\boldsymbol{\lambda}(i) = \left(\mathbf{X}^{\mathrm{T}}(i)\,\mathbf{X}(i)\right)^{-1}\boldsymbol{e}(i) \tag{2-28}$$

Introducing the step-size μ and adding a small constant ε to avoid numerical problems in the matrix inversion, we obtain a more general update equation for the AP algorithm described by

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \mu\mathbf{X}(i)\left(\mathbf{X}^{\mathrm{T}}(i)\,\mathbf{X}(i) + \varepsilon\mathbf{I}_K\right)^{-1}\boldsymbol{e}(i). \tag{2-29}$$

2.3 Sparsity-Aware Adaptive Algorithms

A system is considered to be sparse if only a few of its elements have non-zero values. A sparse signal can be represented as a vector of a finite-dimensional space which can be expressed as a linear combination of a small number of basis vectors of the related space. There are many applications, such as echo cancellation, channel equalization and system identification, where sparse signals and systems are found. However, traditional adaptive algorithms do not exploit the sparsity of the model. When dealing with learning problems, we attempt to extract as much useful information as we can from the system to obtain better results.Sparsity-aware methods encourage sparse coefficients so that the sparsity of the model can be exploited. Under this scope, the sparsity of systems has been the focus of many research works aiming to improve the performance of adaptive algorithms [17–24].

One of the first approaches used to exploit sparsity was the proportionate family of algorithms. This group of algorithms assigns proportional step-sizes to different weights depending on their magnitudes. These algorithms include the proportionate normalized LMS (PNLMS) [25] and the improved PNLMS (IPNLMS) [26]. Since then, several versions of proportionate algorithms were proposed such as the μ-law PNLMS (MPNLMS) [27] and the improved MPNLMS (IMPNLMS) [28] algorithms. In [29] an individual activation factor PNLMS (IAF-PNLMS) algorithm was presented to better distribute the adaptation energy over the coefficients. Additionally, the set-membership

PNLMS (SM-PNLMS) [30], which is a data-selective version of the PNLMS algorithm, was developed. Proportionate algorithms were also extended to the AP algorithm, giving rise to the proportionate AP (PAP) [31] and the improved PAP (IPAP) [32] algorithms. The main advantage of these algorithms is that they accelerate the speed of convergence by reusing multiple past inputs as a single input. Moreover, a data-selective version, the set-membership PAP (SM-PAP) algorithm was introduced in [33].

In recent years, another approach to deal with sparsity based on penalty functions has been adopted. In this context, a penalty function is added to the cost function to take into account the sparsity of the model and then a gradient-based algorithm is derived. Basically, the penalty function attracts the coefficients to zero to improve the convergence speed. In [34], the zero-attracting LMS (ZA-LMS) and the reweighted zero-attracting LSM (RZA-LMS) are presented and used for sparse system identification. This idea was extended to the AP algorithm in [35], where the zero-attracting AP (ZA-AP) and the reweighted zero-attracting AP (RZA-AP) algorithms are introduced. Another example of this kind of algorithm is the zero-attracting recursive least squares (ZA-RLS) algorithm [36]. Other versions of the regularized RLS algorithm are found in [37]. There are also data-selective versions of adaptive algorithms that incorporate a penalty function [38, 39]. A review of common penalty functions used in the literature and another scheme to treat sparsity is addressed in [40]. In general, adaptive algorithms that use a penalty function are computationally less expensive and they also achieve a better trade-off between performance and complexity [41] than proportionate algorithms. However, setting an appropriate value of the regularization term is a critical step in this kind of algorithms, because it directly affects the overall performance of the algorithms.

2.3.1 Sparsity-Aware NLMS

By incorporating a penalty function in the cost function to exploit the sparsity of the system, the optimization problem can be expressed by

$$\min_{\mathbf{w}(i+1)} \frac{1}{2} \parallel \mathbf{w}\left(i+1\right) - \mathbf{w}\left(i\right) \parallel^2 + \alpha f_l\left(\mathbf{w}\left(i+1\right)\right)$$

$$\text{subject to}$$

$$d\left(i\right) - \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right) = 0, \tag{2-30}$$

where f_l is a general penalty function and α is the regularization term that controls the desired penalty. A typical choice for the penalty function is an

l^p norm because of their inherent property to attract the coefficients to zero. Using the Lagrange multipliers to solve the problem, we get

$$\begin{aligned}\mathcal{L}\left(\mathbf{w}\left(i+1\right),\lambda\right) =& \frac{1}{2} \parallel \mathbf{w}\left(i+1\right) - \mathbf{w}\left(i\right) \parallel^2 + \alpha f_l\left(\mathbf{w}\left(i+1\right)\right) \\ &+ \lambda\left(d\left(i\right) - \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right)\right). \end{aligned} \tag{2-31}$$

Taking the gradient with respect to $\mathbf{w}\left(i+1\right)$ and λ of equation (2-31), we have

$$\frac{\partial\mathcal{L}\left(\mathbf{w}\left(i+1\right),\lambda\right)}{\partial\mathbf{w}\left(i+1\right)} = \left(\mathbf{w}\left(i+1\right) - \mathbf{w}\left(i\right)\right) - \lambda\mathbf{x}\left(i\right) + \alpha\frac{\partial f_l\left(\mathbf{w}\left(i+1\right)\right)}{\partial\mathbf{w}\left(i+1\right)}, \tag{2-32}$$

$$\frac{\partial\mathcal{L}\left(\mathbf{w}\left(i\right),\lambda\right)}{\partial\lambda} = d\left(i\right) - \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right). \tag{2-33}$$

Equating the resulting terms to zero leads to:

$$\mathbf{w}\left(i+1\right) = \mathbf{w}\left(i\right) + \lambda\mathbf{x}\left(i\right) - \alpha\frac{\partial f_l\left(\mathbf{w}\left(i+1\right)\right)}{\partial\mathbf{w}\left(i+1\right)} \tag{2-34}$$

$$d\left(i\right) = \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right). \tag{2-35}$$

By premultiplying equation (2-34) by $\mathbf{x}^{\mathrm{T}}\left(i\right)$, we obtain

$$d\left(i\right) = \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i\right) + \lambda\mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{x}\left(i\right) - \alpha\mathbf{x}^{\mathrm{T}}\left(i\right)\frac{\partial f_l\left(\mathbf{w}\left(i+1\right)\right)}{\partial\mathbf{w}\left(i+1\right)}. \tag{2-36}$$

Solving the last equation for λ, we get

$$\lambda \parallel \mathbf{x}\left(i\right) \parallel^2 = d\left(i\right) - \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i\right) + \alpha\mathbf{x}^{\mathrm{T}}\left(i\right)\frac{\partial f_l\left(\mathbf{w}\left(i+1\right)\right)}{\partial\mathbf{w}\left(i+1\right)} \tag{2-37}$$

$$= e\left(i\right) + \alpha\mathbf{x}^{\mathrm{T}}\left(i\right)\frac{\partial f_l\left(\mathbf{w}\left(i+1\right)\right)}{\partial\mathbf{w}\left(i+1\right)} \tag{2-38}$$

$$\lambda = \frac{e\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2} + \frac{\alpha}{\parallel \mathbf{x}\left(i\right) \parallel^2}\mathbf{x}^{\mathrm{T}}\left(i\right)\frac{\partial f_l\left(\mathbf{w}\left(i+1\right)\right)}{\partial\mathbf{w}\left(i+1\right)}. \tag{2-39}$$

Let us assume that $f_l\left(\mathbf{w}\left(i+1\right)\right) \approx f_l\left(\mathbf{w}\left(i\right)\right)$. By including the step-size and substituting λ in equation (2-34), we can finally get the update recursion for the sparsity-aware NLMS algorithm with an arbitrary penalty function [35]:

$$\mathbf{w}\left(i+1\right) = \mathbf{w}\left(i\right) + \frac{\mu e\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2}\mathbf{x}\left(i\right) + \rho\left[\frac{\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2} - \mathbf{I}_N\right]\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}, \tag{2-40}$$

where $\rho = \mu\alpha$.

2.3.2 Sparsity-Aware SM-NLMS

This family of algorithms is a data-selective version of the NLMS family. Let us consider equation (2-30) with a modified constraint:

$$\min_{\mathbf{w}(i+1)} \parallel \mathbf{w}\left(i+1\right) - \mathbf{w}\left(i\right) \parallel^2 + \alpha f_l\left(\mathbf{w}\left(i+1\right)\right)$$
$$\text{subject to}$$
$$\left| d\left(i\right) - \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right)\right| = \gamma \tag{2-41}$$

where γ is the error bound. Solving the equation with the method of Lagrange multipliers and considering the absolute value as two isolate cases, we get

$$\begin{aligned}\mathcal{L}\left(\mathbf{w}\left(i\right),\lambda\right) = &\parallel \mathbf{w}\left(i+1\right) - \mathbf{w}\left(i\right) \parallel^2 + \alpha f_l\left(\mathbf{w}\left(i+1\right)\right) \\ &+ \lambda\left(d\left(i\right) - \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right) - \gamma\right) \qquad (2\text{-}42)\\ \mathcal{L}\left(\mathbf{w}\left(i\right),\lambda\right) = &\parallel \mathbf{w}\left(i+1\right) - \mathbf{w}\left(i\right) \parallel^2 + \alpha f_l\left(\mathbf{w}\left(i+1\right)\right) \\ &+ \lambda\left(d\left(i\right) - \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i+1\right) + \gamma\right) \qquad (2\text{-}43)\end{aligned}$$

Taking the gradient with respect to $\mathbf{w}\left(i+1\right)$ from equation (2-42) we arrive at the same result as in equation (2-34). Premultiplying this equation by $\mathbf{x}^{\mathrm{T}}\left(i\right)$, we obtain

$$d\left(i\right) - \gamma = \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{w}\left(i\right) + \frac{\lambda}{2}\mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{x}\left(i\right) - \frac{\alpha}{2}\mathbf{x}^{\mathrm{T}}\left(i\right)\frac{\partial f_l\left(\mathbf{w}\left(i+1\right)\right)}{\partial \mathbf{w}\left(i+1\right)}. \tag{2-44}$$

Solving for λ leads to :

$$\frac{\lambda}{2} = \frac{\left(e\left(i\right) - \gamma\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2} + \frac{\alpha}{2 \parallel \mathbf{x}\left(i\right) \parallel^2}\mathbf{x}^{\mathrm{T}}\left(i\right)\frac{\partial f_l\left(\mathbf{w}\left(i+1\right)\right)}{\partial \mathbf{w}\left(i+1\right)}. \tag{2-45}$$

Replacing λ in equation (2-34), we have

$$\begin{aligned}\mathbf{w}\left(i+1\right) = &\mathbf{w}\left(i\right) + \mathbf{x}\left(i\right)\left(\mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{x}\left(i\right)\right)^{-1}\left(e\left(i\right) - \gamma\right) \\ &+ \frac{\alpha}{2}\left[\frac{\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2} - \mathbf{I}_N\right]\frac{\partial f_l\left(\mathbf{w}\left(i+1\right)\right)}{\partial \mathbf{w}\left(i+1\right)}. \qquad (2\text{-}46)\end{aligned}$$

Taking into account that the constraint originated two equations, which depend on the value of the error, which means that the adaptation only occurs when the bound constraint is satisfied, we can rewrite (2-46) as one single

equation as follows:

$$\begin{aligned}\mathbf{w}\left(i+1\right) =&\mathbf{w}\left(i\right)+\mathbf{x}\left(i\right)\frac{\left(e\left(i\right)-\gamma\mathrm{sgn}\left(e\left(i\right)\right)\right)}{\parallel\mathbf{x}\left(i\right)\parallel^{2}}\\&+\frac{\alpha}{2}\left[\frac{\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel\mathbf{x}\left(i\right)\parallel^{2}}-\mathbf{I}_{N}\right]\frac{\partial f_{l}\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)},\end{aligned}\tag{2-47}$$

From the above equation we can finally get the update recursion for the sparsity-aware SM-NLMS algorithm, which is a particular case from [38], and can be expressed by

$$\begin{aligned}\mathbf{w}\left(i+1\right) =&\mathbf{w}\left(i\right)+\mu\left(i\right)\frac{e\left(i\right)}{\parallel\mathbf{x}\left(i\right)\parallel^{2}}\mathbf{x}\left(i\right)\\&+\rho\left[\frac{\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel\mathbf{x}\left(i\right)\parallel^{2}}-\mathbf{I}_{N}\right]\frac{\partial f_{l}\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}\end{aligned}\tag{2-48}$$

where $\mu\left(i\right)$ is the adaptive step-size of the algorithm defined by

$$\mu\left(i\right)=\begin{cases}\left(1-\frac{\gamma}{|e(i)|}\right) & |e\left(i\right)|>\gamma\\0 & \text{otherwise}\end{cases}\tag{2-49}$$

2.4 Kernel Adaptive Filtering (KAF)

As mentioned in previous sections, conventional adaptive algorithms work with linear structures, limiting the performance that they can achieve and constraining the number of problems that can be solved. Under this scope, a new family of nonlinear adaptive filtering algorithms based on kernels has been developed [13]. The main objective of these algorithms is to learn an arbitrary input-output mapping $f:\mathbb{F}\rightarrow\mathbb{R}$ based on a sequence of samples and a kernel. Basically, a kernel $\kappa\left(\cdot,\cdot\right)$ is a function that generally returns a real number. This number measures the similarity between two inputs.

One of the main advantages of KAF algorithms is that they are universal approximators [13], which gives them the capability to address complex and nonlinear problems. In other words, they can in principle model any input-output mapping. Most of these algorithms have been designed to solve convex optimization problems, avoiding local minima problems, which is also a desirable characteristic. However, the computational complexity is significantly higher than their linear counterparts.

To compute the mapping we use the following rule:

$$f_{i}=f_{i-1}+gain\left(i\right)e\left(i\right),\tag{2-50}$$

where the actual mapping estimate depends on the previous mapping estimate and also on a correcting term that is proportional to the prediction error. The gain function and the error of equation (2-50) fully describe the different algorithms.

The KAF algorithms map the data to a high-dimensional space through kernels. Then, linear methods can be applied on the transformed data. Let us consider a simple case of two dimensional vectors belonging to two different classes, that are not linearly separable in the two-dimensional space. Mapping the data to a convenient high dimensional space transforms the problem into a linearly separable problem. Figure 2.2 describes this idea, where two-dimensional vectors are being mapped to a three dimensional space.

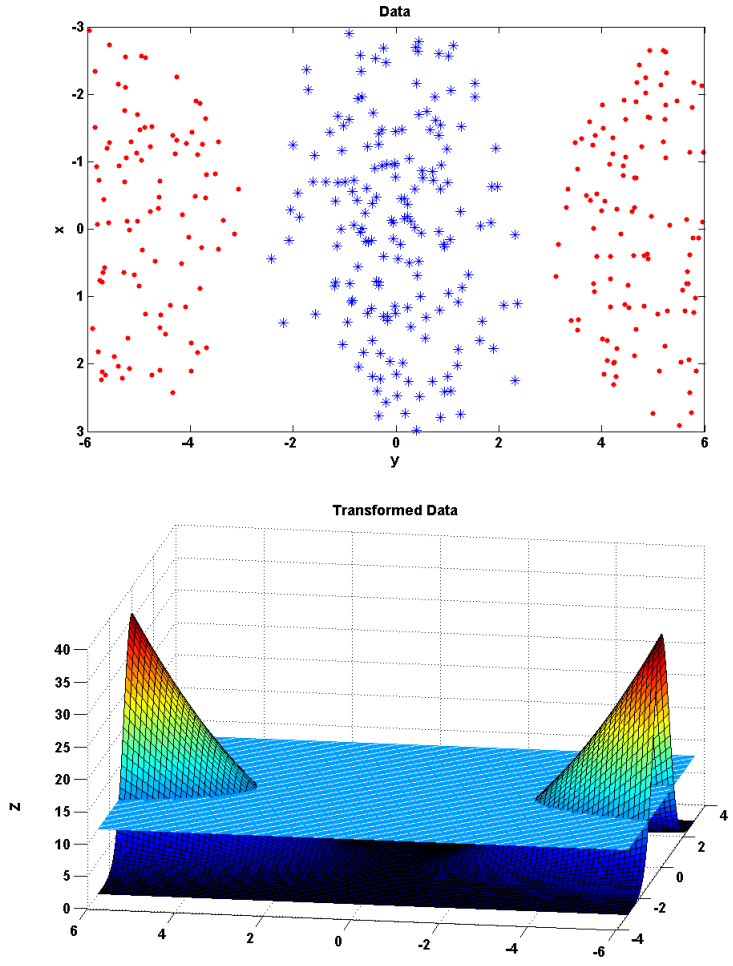

Figure 2.2: Two-dimensional data mapped into a three-dimensional space

Basically a nonlinear problem is transformed into a linear one. The relevant point about implementing kernel functions is that the scalar product can be implicitly computed in the feature space by a kernel evaluation, without explicitly using or even knowing the mapping applied to the data [42]. This

means that there is no need to perform any operation on the high dimensional space, as long as the terms of the algorithm are expressed as an inner product. This methodology is known as the "kernel trick" and allows us to compute scalar products in spaces, where the computations are hard to perform. As a result, we avoid a significant increase of the computational complexity, which is one of the major problems that arises when working with high dimensional spaces. In particular we have

$$\kappa\left(\mathbf{x}, \mathbf{x}'\right) = \left\langle \kappa\left(\cdot, \mathbf{x}\right), \kappa\left(\cdot, \mathbf{x}'\right)\right\rangle . \tag{2-51}$$

Every algorithm that only uses scalar products can be implicitly implemented in the feature space by using kernels [42]. To summarize, we need to reformulate the classical adaptive algorithms in terms of the inner product of the mapped inputs to elegantly create nonlinear structures.

Several kernel functions are described in the literature [13]. Choosing a kernel function is equivalent to implicitly defining a feature space where the algorithms are performed. Let us now introduce two commonly used kernel functions. The first one is the Gaussian kernel, defined by

$$\kappa\left(\mathbf{x}, \mathbf{x}'\right) = \exp\left(-\frac{\| \mathbf{x} - \mathbf{x}' \|^2}{2\nu^2}\right), \tag{2-52}$$

where ν is the kernel bandwidth that specifies the shape of the kernel function. Another important kernel function is the polynomial kernel, given by

$$\kappa\left(\mathbf{x}, \mathbf{x}'\right) = \left(\mathbf{x}^{\mathrm{T}}\mathbf{x}' + 1\right)^{p}, \tag{2-53}$$

with $p \in \mathbb{N}$ known as the polynomial degree.

All KAF algorithms have to deal with kernel expansions. In other words, they create a growing structure, also called dictionary, where they keep every new data input that arrives to compute the estimate of the desired output. The natural problem that arises is that the time and computational cost required to compute a certain output could exceed the tolerable limits for a specific application. Several criteria were proposed to solve this problem such as algorithms with fixed dictionary size [43, 44]. One of the most simple criteria is the novelty criterion (NC), presented in [45]. Basically it establishes two thresholds to limit the size of the dictionary. Another method, the approximate linear dependency (ALD) was proposed in [46] and verifies if a new input can be expressed as a linear combination of the elements stored, before adding this input to the dictionary. The coherence criterion (CC) was introduced in [47,48] also to limit the size of the dictionary based on the similarity of the inputs. A measure called surprise (SC) was presented in [49] to remove redundant data from the dictionary.

2.4.1 Kernel Least-Mean-Squares (KLMS)

This algorithm was proposed in [50] and is widely used because of its simplicity. Let us consider an adaptive filtering problem with a sequence of training samples given by $\{\mathbf{x}(i), d(i)\}$, where $\mathbf{x}(i)$ is the N-dimensional input vector of the system and $d(i)$ represents the desired signal at time instant i. The output of the adaptive filter is given by equation (2-1). Let us define a non-linear transformation denoted by $\varphi : \mathbb{R}^n \to \mathbb{F}$ that maps the input to a high-dimensional feature space. Applying the transformation stated before, we map the input and the weights to a high-dimensional space obtaining:

$$\boldsymbol{\varphi}(i) = \varphi(\mathbf{x}(i)) \tag{2-54}$$

$$\boldsymbol{\omega}(i) = \varphi(\mathbf{w}(i)) \tag{2-55}$$

The error generated by the system is given by $e(i) = d(i) - \boldsymbol{\omega}^{\mathrm{T}}(i)\,\boldsymbol{\varphi}(i)$. The main objective of the kernel-based adaptive algorithms is to implement an input-output mapping, such that the mean square error generated by the system is minimized.

Using the equations of the LMS algorithm over the sequence $\{\boldsymbol{\varphi}(i), d(i)\}$, we have

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \mu e(i)\boldsymbol{\varphi}(i) \tag{2-56}$$

Let us now express the last equation in a recursive manner as follows:

$$\begin{aligned}
\boldsymbol{\omega}(i+1) &= \boldsymbol{\omega}(i) + \mu e(i)\boldsymbol{\varphi}(i) \\
&= \boldsymbol{\omega}(i-1) + \mu e(i-1)\boldsymbol{\varphi}(i-1) + \mu e(i)\boldsymbol{\varphi}(i) \\
&= \boldsymbol{\omega}(i-1) + \mu\left[e(i-1)\boldsymbol{\varphi}(i-1) + e(i)\boldsymbol{\varphi}(i)\right] \\
&\quad\vdots \\
&= \boldsymbol{\omega}(0) + \mu\sum_{j=1}^{i} e(j)\boldsymbol{\varphi}(j)
\end{aligned} \tag{2-57}$$

Let us assume that the initial value of the weights is $\boldsymbol{\omega}(0) = 0$. The weight estimate is now given by

$$\boldsymbol{\omega}(i+1) = \mu\sum_{j=1}^{i} e(j)\boldsymbol{\varphi}(j) \tag{2-58}$$

This equation expresses the weights estimate as a linear combination of all the past inputs. The system output to a new input vector $\mathbf{x}'$ is given by

$$\begin{aligned}\boldsymbol{\omega}(i+1)^{\mathrm{T}}\varphi(\mathbf{x}') &= \left[\mu\sum_{j=1}^{i} e(j)\boldsymbol{\varphi}(j)^{\mathrm{T}}\right]\varphi(\mathbf{x}') \\ &= \mu\sum_{j=1}^{i} e(j)\left[\boldsymbol{\varphi}(j)^{\mathrm{T}}\varphi(\mathbf{x}')\right] \\ &= \mu\sum_{j=1}^{i} e(j)\kappa(\mathbf{x}\,(j)^{\mathrm{T}},\mathbf{x}') \end{aligned} \tag{2-59}$$

In this kernelized version of the LMS algorithm, the filtering process is performed just by evaluating the kernel of the input vectors. The algorithm uses the sum of the past errors multiplied by the kernel evaluation of the past inputs to obtain the estimated output $y\,(i)$. The inputs are stored in matrix $\mathcal{C}\,(i)$, also known as the dictionary, for future calculations. The past errors multiplied by the step-size create the coefficients vector expressed by

$$\boldsymbol{a}\,(i) = \mu e\,(i)\,. \tag{2-60}$$

The KLMS algorithm is summarized in Algorithm 1.

Algorithm 1 Kernel Least Mean Squares

Initialization

1.Choose μ, and κ.

2.$\boldsymbol{C}\,(1) = \{\mathbf{x}\,(1)\}$

3.$a_1\,(1) = \mu\,(1)\,d\,(1)$

Computation

4.**while** $\{\mathbf{x}\,(i)\,, d\,(i)\}$ available **do**:

%Compute the output

5. $f_{i-1}(\mathbf{x}\,(i)) = \sum_{k=1}^{i-1} a_k\,(i)\,\kappa\,(\mathbf{x}\,(i)\,,\boldsymbol{c}_k)$

%Compute the error

6. $e\,(i) = d\,(i) - f_{i-1}(\mathbf{x}\,(i))$

%Update the coefficients

7. $\boldsymbol{a}\,(i+1) = \begin{bmatrix} \boldsymbol{a}\,(i) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mu\,(i)\,e\,(i) \end{bmatrix}$

%Store the new center

8. $\boldsymbol{\mathcal{C}}\,(i+1) = \{\boldsymbol{\mathcal{C}}\,(i)\,, \mathbf{x}\,(i)\}$

9.**end while**

2.4.2
Kernel Affine Projection (KAP) Algorithm

The AP algorithms can be extended to an appropriate kernelized version creating the KAP algorithms. The KAP algorithms were first proposed in [51]

giving rise to four different algorithms. In this project we focus on the KAP-2 algorithm. An analysis of the other three algorithms can be found in [13, 51].

Applying the transformation $\varphi : \mathbb{R}^n \to \mathbb{F}$, to equation (2-17), we obtain

$$\boldsymbol{\Phi}(i) = \varphi(\mathbf{X}(i)) = \left[\begin{array}{cccc} \boldsymbol{\varphi}(i) & \boldsymbol{\varphi}(i-1) & \cdots & \boldsymbol{\varphi}(i-K+1) \end{array} \right]. \tag{2-61}$$

This leads to a modification in equation (2-18) which is given by

$$\boldsymbol{y}(i) = \boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\omega}(i) = \left[\begin{array}{c} y_0(i) \\ y_1(i) \\ \vdots \\ y_{K-1}(i) \end{array} \right], \tag{2-62}$$

remaining equations (2-19) and (2-20) without changes. Now we need to solve the following constrained optimization:

$$\begin{array}{c} \min\limits_{\boldsymbol{\omega}(i+1)} \ \dfrac{1}{2} \parallel \boldsymbol{\omega}(i+1) - \boldsymbol{\omega}(i) \parallel^2 \\ \text{subject to} \\ \boldsymbol{d}(i) - \boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\omega}(i+1) = \mathbf{0} \end{array} \tag{2-63}$$

Using the method of Lagrange multipliers to solve equation (2-63), we obtain:

$$\mathcal{L}(\boldsymbol{\omega}(i+1)) = \frac{1}{2} \parallel \boldsymbol{\omega}(i+1) - \boldsymbol{\omega}(i) \parallel^2 + \boldsymbol{\lambda}^{\mathrm{T}}(i) \left(\boldsymbol{d}(i) - \boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\omega}(i+1) \right). \tag{2-64}$$

Applying the same procedure as in the AP algorithm we arrive at the KAP algorithm update equation:

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \mu \boldsymbol{\Phi}(i) \left(\boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\Phi}(i) + \epsilon \mathbf{I} \right)^{-1} \boldsymbol{e}(i), \tag{2-65}$$

which is consistent with the results obtained in [51]. We usually do not have access to the vector $\boldsymbol{\omega}(i+1)$. Therefore we need to reformulate equation (2-65) in terms of input vector inner products. For this purpose, we follow a similar development as in [13] and [52] , where the KAP-1 and the complex KAP algorithms were presented, respectively. Consider the case where the initial value of $\boldsymbol{\omega}$ is zero. We can now calculate the values of $\boldsymbol{\omega}$ recursively using (2-65) as follows:

$$\begin{aligned}\boldsymbol{\omega}(1) &= \boldsymbol{\omega}(0) + \mu\boldsymbol{\Phi}(1)\left(\boldsymbol{\Phi}^{\mathrm{T}}(1)\boldsymbol{\Phi}(1) + \epsilon\mathbf{I}\right)^{-1}\boldsymbol{e}(1) \\ &= \mu\boldsymbol{\Phi}(1)\left(\boldsymbol{\Phi}^{\mathrm{T}}(1)\boldsymbol{\Phi}(1) + \epsilon\mathbf{I}\right)^{-1}\left(d(1) - \boldsymbol{\Phi}^{\mathrm{T}}(1)\boldsymbol{\omega}(0)\right) \\ &= \mu\boldsymbol{\varphi}(1)\left(\boldsymbol{\varphi}^{\mathrm{T}}(1)\boldsymbol{\varphi}(1) + \epsilon\mathbf{I}\right)^{-1} d(1) \\ &= \left(\mu d(1) < \boldsymbol{\varphi}^{\mathrm{T}}(1), \boldsymbol{\varphi}(1) >^{-1}\right)\boldsymbol{\varphi}(1) \\ &= a_1(1)\boldsymbol{\varphi}(1). \end{aligned} \tag{2-66}$$

For the second iteration, we have

$$\begin{aligned}\boldsymbol{\omega}(2) &= \boldsymbol{\omega}(1) + \mu\boldsymbol{\Phi}(2)\left(\boldsymbol{\Phi}^{\mathrm{T}}(2)\boldsymbol{\Phi}(2) + \epsilon\mathbf{I}\right)^{-1}\boldsymbol{e}(2) \\ &= a_1(1)\boldsymbol{\varphi}(1) + \mu\boldsymbol{\Phi}(2)\tilde{\mathbf{a}}(2), \end{aligned} \tag{2-67}$$

where the vector $\tilde{\mathbf{a}}(i)$ is defined by

$$\tilde{\mathbf{a}}(i) = \left(\boldsymbol{\Phi}^{\mathrm{T}}(i)\boldsymbol{\Phi}(i) + \epsilon\mathbf{I}\right)^{-1}\boldsymbol{e}(i). \tag{2-68}$$

Expanding the terms of vector $\tilde{\mathbf{a}}(i)$, we get

$$\begin{aligned}\boldsymbol{\omega}(2) &= a_1(1)\boldsymbol{\varphi}(1) + \mu\begin{bmatrix}\boldsymbol{\varphi}(1) & \boldsymbol{\varphi}(2)\end{bmatrix}\begin{bmatrix}\tilde{a}_1(2) & \tilde{a}_2(2)\end{bmatrix}^{\mathrm{T}} \\ &= a_1(1)\boldsymbol{\varphi}(1) + \mu\tilde{a}_1(2)\boldsymbol{\varphi}(1) + \mu\tilde{a}_2(2)\boldsymbol{\varphi}(2) \\ &= (a_1(1) + \mu\tilde{a}_1(2))\boldsymbol{\varphi}(1) + \mu\tilde{a}_2(2)\boldsymbol{\varphi}(2) \\ &= a_1(2)\boldsymbol{\varphi}(1) + a_2(2)\boldsymbol{\varphi}(2). \end{aligned} \tag{2-69}$$

By the analysis of equations (2-66) and (2-69) and also based on the Representer Theorem [53] we can state that

$$\boldsymbol{\omega}(i) = \sum_{j=1}^{i} a_j(i)\boldsymbol{\varphi}(j) \tag{2-70}$$

Substituting equations (2-68) and (2-70) in the update equation (2-65), we obtain the following expression:

$$\boldsymbol{\omega}(i+1) = \sum_{j=1}^{i} a_j(i)\boldsymbol{\varphi}(j) + \mu\boldsymbol{\Phi}(i)\tilde{\mathbf{a}}(i). \tag{2-71}$$

Expanding the terms of vector $\tilde{\mathbf{a}}(i)$ as before results in

$$\boldsymbol{\omega}(i+1) = \sum_{j=1}^{i} a_j(i)\boldsymbol{\varphi}(j) + \mu\sum_{k=i-K+2}^{i+1}\tilde{a}_{k+K-i-2}(i)\boldsymbol{\varphi}(k). \tag{2-72}$$

From the last equation we find that at every iteration one element is added to the coefficient vector $\boldsymbol{a}(i)$. We can also see that only the last K coefficients are modified at every iteration, whereas the other coefficients

remain the same. Consequently, we can summarize the coefficients updates as follows:

$$a_k(i+1) = \begin{cases} \mu\tilde{a}_k(i), & k = i+1 \\ a_k(i) + \mu\tilde{a}_{K+k-i-2}(i), & i-K+2 \leq k \leq i \\ a_k(i) & 1 \leq k < i-K+1 \end{cases} \tag{2-73}$$

The output of the filter is given by

$$\begin{aligned} \boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\omega}(i) &= \sum_{j=1}^{i} a_j(i)\,\boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\varphi}(j) \\ &= \sum_{j=1}^{i} a_j(i)\left[\begin{array}{ccc} \boldsymbol{\varphi}(i) & \cdots & \boldsymbol{\varphi}(i-K+1) \end{array}\right]^{\mathrm{T}}\boldsymbol{\varphi}(j) \end{aligned} \tag{2-74}$$

Expanding the terms of the last equation, we get

$$\begin{aligned} \boldsymbol{y}(i) &= \left[\begin{array}{c} \sum_{j=1}^{i} a_j(i)\,\boldsymbol{\varphi}^{\mathrm{T}}(i)\,\boldsymbol{\varphi}(j) \\ \sum_{j=1}^{i} a_j(i)\,\boldsymbol{\varphi}^{\mathrm{T}}(i-1)\,\boldsymbol{\varphi}(j) \\ \vdots \\ \sum_{j=1}^{i} a_j(i)\,\boldsymbol{\varphi}^{\mathrm{T}}(i-K+1)\,\boldsymbol{\varphi}(j) \end{array}\right] \\ &= \left[\begin{array}{c} \sum_{j=1}^{i} a_j(i)\,\kappa(\mathbf{x}(i),\mathbf{x}(j)) \\ \sum_{j=1}^{i} a_j(i)\,\kappa(\mathbf{x}(i-1),\mathbf{x}(j)) \\ \vdots \\ \sum_{j=1}^{i} a_j(i)\,\kappa(\mathbf{x}(i-K+1),\mathbf{x}(j)) \end{array}\right]. \end{aligned} \tag{2-75}$$

Finally, we can compute the error using equation (2-20). Equations (2-68), (2-73), (2-75) and (2-20) fully describe the KAP algorithm, which is summarized in Algorithm 2. In general, kernel-based algorithms deal with models whose order equals the size of the training set. Therefore, implementing a methodology to control the increase in the model order is still the focus of many research works.

Algorithm 2 Kernel Affine Projection (KAP-2) Algorithm

Initialization

1.Fix the step-size μ, the regularization factor ϵ and the parameter K

2.$a_1(1) = \mu d(1)$

Computation

3.**while** $\{\mathbf{x}(i), d(i)\}$ available **do**:

%Insert a new element into the coefficients vector $\boldsymbol{a}(i-1)$

4. $a_i(i-1) = 0$

5. **for** $n = 1$ to $\min(i, K)$ **do**

6. **for** $m = 1$ to $\min(i, K)$ **do**

%Compute the $K \times K$ matrix $\tilde{\mathbf{A}}$

7. $\tilde{A}_{n,m} = \kappa(\mathbf{x}(i-n+1), \mathbf{x}(i-m+1))$

8. **end for**

9. **end for**

%Compute the inverse of matrix $\tilde{\mathbf{A}}$

10. $\tilde{\mathbf{A}} = (\tilde{\mathbf{A}} + \epsilon \mathbf{I})^{-1}$

11. **for** $k = 1$ to $\min(i, K)$ **do**

%Evaluate the outputs

12. $y_k(i) = \sum_{j=1}^{i} a_j(i) \kappa(\mathbf{x}(i-k+1), \mathbf{x}(j))$

%Evaluate the errors

13. $e_k(i) = d(i-k+1) - y_k(i)$

14. **end for**

%Compute the update for the coefficients

15. $\tilde{\mathbf{a}} = \tilde{\mathbf{A}}\mathbf{e}(i)$

16. **for** $k = 1$ to $\min(i, K)$ **do**

%Update the $\min(i, K)$ most recent units

17. $a_{i-k+1}(i) = a_{i-k+1}(i-1) + \mu \tilde{a}_{i-k+1}$

18. **end for**

13.**end while**

3
Sparsity-Aware Data-Selective Adaptive Algorithms

In this chapter we introduce a new framework for deriving sparsity-aware set-membership adaptive algorithms with adjustable penalties using arbitrary penalty functions. Then, we derive the proposed sparsity-aware set-membership algorithms with adjustable penalties based on a gradient descent approach. We also develop a statistical analysis of adaptive algorithms with arbitrary penalty functions to describe the steady-state performance of these algorithms, thereby addressing two problems of sparsity-aware algorithms, namely, the exploitation of sparsity in the updates and in the parameter vector.

3.1
Derivation Framework

Let us consider a gradient descent approach, where our model is updated by the recursive equation defined by

$$\mathbf{w}\left(i+1\right)=\mathbf{w}\left(i\right)-\mu\left(i\right)\frac{\partial J\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}. \tag{3-1}$$

Now that the update recursion has been established, let us define a mean-square error cost function with a general penalty function as described by

$$J\left(\mathbf{w}\left(i\right)\right)=\frac{1}{2}\mathbb{E}\left[e\left(i\right)^{2}\right]+\alpha\left(i\right)f_{l}\left(\mathbf{w}\left(i\right)\right), \tag{3-2}$$

where the function $f_{l}\left(\mathbf{w}\left(i\right)\right)$ is a general penalty function used to improve the performance of adaptive algorithms in the presence of sparsity and $\alpha(i)$ is a regularization term that imposes and controls the desired penalty. The cost function can be rewritten as follows:

$$J\left(\mathbf{w}\left(i\right)\right)=\frac{1}{2}\mathbb{E}\left[\left|d\left(i\right)-\mathbf{w}^{\mathrm{T}}\left(i\right)\mathbf{x}\left(i\right)\right|^{2}\right]+\alpha\left(i\right)f_{l}\left(\mathbf{w}\left(i\right)\right). \tag{3-3}$$

Taking the instantaneous gradient of the cost function with respect to $\mathbf{w}\left(i\right)$, we obtain

$$\frac{\partial J\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}=-e\left(i\right)\mathbf{x}\left(i\right)+\alpha\left(i\right)\frac{\partial f_{l}\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}. \tag{3-4}$$

Replacing the last result in the update equation (3-1), we get

$$\mathbf{w}\left(i+1\right)=\mathbf{w}\left(i\right)-\mu\left(i\right)\left[-e\left(i\right)\mathbf{x}\left(i\right)+\alpha\left(i\right)\frac{\partial f_{l}\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}\right]. \tag{3-5}$$

Note that we employ a time index in μ to designate a variable step-size. Following the SM-NLMS approach, the updates are performed only if $|e(i)| > \gamma$, which leads to the general equation to update the weights:

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \mu(i)\, e(i)\, \mathbf{x}(i) - \rho(i) \frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}, \tag{3-6}$$

where $\rho(i) = \mu(i)\,\alpha(i)$. Using the equality constraint, i.e., the *a posteriori* error $|e_p(i)| = \gamma$, we have

$$|d(i) - \mathbf{w}^{\mathrm{T}}(i+1)\,\mathbf{x}(i)| = \gamma. \tag{3-7}$$

Substituting (3-6) in (3-7) and solving for γ yields

$$\gamma = \left| d(i) - \mathbf{w}^{\mathrm{T}}(i)\,\mathbf{x}(i) - \left(\mu(i)\, e(i)\, \mathbf{x}(i) - \rho(i) \frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right)^{\mathrm{T}} \mathbf{x}(i) \right|. \tag{3-8}$$

Multiplying both sides of the last equation by $\frac{e_p(i)}{|e_p(i)|}$, we obtain

$$\begin{aligned} \gamma \frac{e_p(i)}{|e_p(i)|} =& d(i) - \mathbf{w}^{\mathrm{T}}(i)\,\mathbf{x}(i) \\ & - \left(\mu(i)\, e(i)\, \mathbf{x}(i) - \rho(i) \frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right)^{\mathrm{T}} \mathbf{x}(i). \end{aligned} \tag{3-9}$$

Simplifying terms, we get

$$\gamma \mathrm{sgn}(e_p(i)) = e(i) - \mu(i)\, e(i)\, \mathbf{x}^{\mathrm{T}}(i)\,\mathbf{x}(i) + \rho(i) \left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]^{\mathrm{T}} \mathbf{x}(i). \tag{3-10}$$

Since the constraint forces that $|e_{ap}(i)| = \gamma$, then the function $\mathrm{sgn}(e_{ap}(i))$ generates two possible equations given by

$$\gamma = e(i) - \mu(i)\, e(i) \parallel \mathbf{x}(i) \parallel^2 + \rho(i) \left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]^{\mathrm{T}} \mathbf{x}(i), \tag{3-11}$$

$$-\gamma = e(i) - \mu(i)\, e(i) \parallel \mathbf{x}(i) \parallel^2 + \rho(i) \left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]^{\mathrm{T}} \mathbf{x}(i). \tag{3-12}$$

We can express equations (3-11) and (3-12) as a single equation as follows:

$$\begin{aligned} e(i)\left(1 - \frac{\gamma}{|e(i)|}\right) =& \mu(i)\, e(i) \parallel \mathbf{x}(i) \parallel^2 \\ & - \alpha(i)\,\mu(i) \left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]^{\mathrm{T}} \mathbf{x}(i), \end{aligned} \tag{3-13}$$

where we take into account that the term $\left(1 + \frac{\gamma}{|e(i)|}\right)$ would produce a growing step-size, leading to a divergent algorithm. Isolating the step-size from the last equation we obtain

$$\mu(i) = \frac{e(i)\left(1 - \frac{\gamma}{|e(i)|}\right)}{\left(e(i) \parallel \mathbf{x}(i) \parallel^2 - \alpha(i)\left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]^{\mathrm{T}} \mathbf{x}(i)\right)} \tag{3-14}$$

We then use equation (3-13) to find $\alpha(i+1)$, recursively, as follows:

$$\mu(i)\,\alpha(i+1)\left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]^{\mathrm{T}} \mathbf{x}(i) = \mu(i)\,e(i) \parallel \mathbf{x}(i) \parallel^2 - e(i)\left(1 - \frac{\gamma}{|e(i)|}\right), \tag{3-15}$$

$$\alpha(i+1) = \frac{e(i)\left(\frac{\gamma}{|e(i)|} + \mu(i) \parallel \mathbf{x}(i) \parallel^2 - 1\right)}{\mu(i)\left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]^{\mathrm{T}} \mathbf{x}(i)}. \tag{3-16}$$

Equations (3-6), (3-14) and (3-16) fully describe the proposed sparsity-aware SM-NLMS algorithm with adjustable penalties. The pseudo-code for the algorithm is presented in Algorithm 3.

Algorithm 3 Sparsity-Aware SM-NLMS Algorithm with Adjustable Penalty Function

---

Initialization

1. Define the penalty function $f_l(\mathbf{w}(i))$
2. Set initial values for γ, $\alpha(0)$ and $\mathbf{w}(0)$

Computation

4. **while** $\{\mathbf{x}(i), d(i)\}$ available **do**:

%Compute the errors

5. $e(i) = d(i) - \mathbf{w}^{\mathbf{T}}(i)\,\mathbf{x}(i)$
6. Compute the gradient $\left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]$

%Compute the step-size

7. $\mu(i) = \frac{e(i)\left(1-\frac{\gamma}{|e(i)|}\right)}{\left(e(i)\|\mathbf{x}(i)\|^2 - \alpha(i)\left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]^{\mathbf{T}}\mathbf{x}(i)\right)}$

%Update the weights

8. $\mathbf{w}(i+1) = \mathbf{w}(i) + \mu(i)\,e(i)\,\mathbf{x}(i) - \mu(i)\,\alpha(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}$

%Update the regularization term

9. $\alpha(i+1) = \frac{e(i)\left(\frac{\gamma}{|e(i)|} + \mu(i)\|\mathbf{x}(i)\|^2 - 1\right)}{\mu(i)\left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]^{\mathrm{T}}\mathbf{x}(i)}$

10. **end while**

We can easily show that if we set $\alpha(i)$ to zero, which means that there is no penalty function being applied, we get the conventional SM-NLMS algorithm. Table 3.1 summarizes the most commonly used penalty functions.

Table 3.1: Penalty Functions

Algorithm	Penalty Function	Gradient
ZA	$f_l(\mathbf{w}) = \parallel \mathbf{w} \parallel_1$	$sgn(\mathbf{w})$
RZA	$f_l(\mathbf{w}) = \sum_{n=1}^{N} ln\left(1 + \frac{\lvert \mathrm{W}_n \rvert}{\varepsilon'}\right)$	$\frac{sgn(\mathbf{w})}{\varepsilon' + \lvert \mathbf{w} \rvert}$
EZA	$f_l(\mathbf{w}) \approx \parallel \mathbf{w} \parallel_0 \approx \sum_{n=1}^{N}\left(1 - e^{-\beta \lvert \mathrm{W}_n \rvert}\right)$	$\beta e^{-\beta \lvert \mathbf{w} \rvert} sgn(\mathbf{w})$

3.1.1 Proposed ZA-SM-NLMS-ADP Algorithm

Substituting the l_1 regularization function into the expression of update recursion of Algorithm 3, we get the new update equations expressed by

$$\mathbf{w}(i) = \mathbf{w}(i-1) + \mu(i)\, e(i)\, \mathbf{x}(i) - \rho(i) \operatorname{sgn}\left[\mathbf{w}(i-1)\right], \tag{3-17}$$

$$\mu(i) = \frac{e(i)\left(1 - \frac{\gamma}{\lvert e(i) \rvert}\right)}{\left(e(i) \parallel \mathbf{x}(i) \parallel^2 - \alpha(i) \operatorname{sgn}^{\mathrm{T}}\left[\mathbf{w}(i-1)\right] \mathbf{x}(i)\right)}. \tag{3-18}$$

The regularization term is given by

$$\alpha(i+1) = \frac{e(i)\left(\frac{\gamma}{\lvert e(i) \rvert} + \mu(i) \parallel \mathbf{x}(i) \parallel^2 - 1\right)}{\mu(i) \operatorname{sgn}^{\mathrm{T}}\left[\mathbf{w}(i-1)\right] \mathbf{x}(i)}. \tag{3-19}$$

3.1.2 Proposed RZA-SM-NLMS-ADP Algorithm

If we use the log-sum penalty function, the recursion of the proposed RZA-SM-NLMS-ADP algorithm becomes:

$$\mathbf{w}(i) = \mathbf{w}(i-1) + \mu(i)\, e(i)\, \mathbf{x}(i) - \rho(i) \left(\frac{\operatorname{sgn}\left[\mathbf{w}(i-1)\right]}{\varepsilon' + \lvert \mathbf{w}(i-1) \rvert}\right). \tag{3-20}$$

The step-size follows the next updating rule:

$$\mu(i) = \frac{e(i)\left(1 - \frac{\gamma}{\lvert e(i) \rvert}\right)}{\left(e(i) \parallel \mathbf{x}(i) \parallel^2 - \alpha(i)\left(\frac{\operatorname{sgn}[\mathbf{w}(i-1)]}{\varepsilon' + \lvert \mathbf{w}^{\mathrm{T}}(i-1) \rvert}\right) \mathbf{x}(i)\right)}, \tag{3-21}$$

and the regularization term can be calculated by

$$\alpha(i+1) = \frac{e(i)\left(\frac{\gamma}{\lvert e(i) \rvert} + \mu(i) \parallel \mathbf{x}(i) \parallel^2 - 1\right)}{\mu(i)\left(\frac{\operatorname{sgn}[\mathbf{w}(i-1)]}{\varepsilon' + \lvert \mathbf{w}^{\mathrm{T}}(i-1) \rvert}\right) \mathbf{x}(i)}. \tag{3-22}$$

3.1.3
Proposed EZA-SM-NLMS-ADP Algorithm

The l_0 regularization requires a computationally expensive search. To avoid this problem we use the approximation presented in Table 3.1, so that the weight-update equation reduces to:

$$\mathbf{w}(i) = \mathbf{w}(i-1) + \mu(i)\, e(i)\, \mathbf{x}(i) - \beta\rho(i)\, e^{-\beta|\mathbf{w}(i)|}\left(sgn\left(\mathbf{w}(i)\right)\right), \tag{3-23}$$

where the step-size and the regularization term are given by

$$\mu(i) = \frac{e(i)\left(1 - \frac{\gamma}{|e(i)|}\right)}{\left(e(i) \parallel \mathbf{x}(i) \parallel^2 - \beta\alpha(i)\, e^{-\beta|\mathbf{w}^{\mathrm{T}}(i-1)|}\left(sgn\left(\mathbf{w}^{\mathrm{T}}(i-1)\right)\right)\mathbf{x}(i)\right)}, \tag{3-24}$$

$$\alpha(i+1) = \frac{e(i)\left(\frac{\gamma}{|e(i)|} + \mu(i) \parallel \mathbf{x}(i) \parallel^2 - 1\right)}{\beta\mu(i)\, e^{-\beta|\mathbf{w}^{\mathrm{T}}(i-1)|}\left(sgn\left(\mathbf{w}^{\mathrm{T}}(i-1)\right)\right)\mathbf{x}(i)}. \tag{3-25}$$

3.2
Statistical analysis

The main parameters used to compare adaptive algorithms, as explained in [54–56], are the mean square error (MSE), the excess mean square error (EMSE), the mean-square deviation (MSD) and the misadjustment (M). The mean square error and the minimum mean square error (MMSE) are defined as

$$MSE \triangleq \mathbb{E}\left[\left(e(i)\right)^2\right], \tag{3-26}$$

$$MMSE \triangleq \lim_{i\to\infty} \mathbb{E}\left[\left(e(i)\right)^2\right]. \tag{3-27}$$

The EMSE and M are, respectively, defined by

$$EMSE \triangleq MSE - MMSE, \tag{3-28}$$

$$M \triangleq \frac{EMSE}{MMSE}. \tag{3-29}$$

Let us denote by N the length of the weights and inputs. For the rest of this section, we consider that the following assumptions hold [57]:

- The input vector $\mathbf{x}(i) \in \mathbb{R}^N$ is random vector whose elements are identically distributed and have zero mean and variance equal to σ_x^2.
- The random variable $n(i)$ represents white Gaussian noise with zero mean and variance equal to σ_n^2 .
- The variables $x_j(i)$ and $n(i)$ are statistically independent for all i, j.

– There exists an optimum model of the weights represented as $\mathbf{w}_o$, so that the desired output is expressed by

$$d\left(i\right) = \mathbf{w}_o^{\mathrm{T}}\mathbf{x}\left(i\right) + n(i). \tag{3-30}$$

Under these conditions, the error defined in equation (2-2) becomes:

$$\begin{aligned} e\left(i\right) &= \mathbf{w}_o^{\mathrm{T}}\mathbf{x}\left(i\right) + n(i) - \mathbf{w}^{\mathrm{T}}\left(i\right)\mathbf{x}\left(i\right) \\ &= \left(\mathbf{w}_o - \mathbf{w}\left(i\right)\right)^{\mathrm{T}}\mathbf{x}\left(i\right) + n(i). \end{aligned} \tag{3-31}$$

Let us now define the a priori error $e_a\left(i\right)$ and the *a posteriori* error $e_p\left(i\right)$, in the absence of observation error, by the following expressions:

$$e_a\left(i\right) = \mathbf{w_o}^{\mathrm{T}}\mathbf{x}\left(i\right) - \mathbf{w}^{\mathrm{T}}\left(i\right)\mathbf{x}\left(i\right) \tag{3-32}$$

$$e_p\left(i\right) = \mathbf{w_o}^{\mathrm{T}}\mathbf{x}\left(i\right) - \mathbf{w}^{\mathrm{T}}\left(i+1\right)\mathbf{x}\left(i\right). \tag{3-33}$$

Replacing equation (3-32) in (3-31), we obtain the following relation:

$$e\left(i\right) = e_a\left(i\right) + n\left(i\right). \tag{3-34}$$

3.2.1 Mean Weight Behavior

Let us now analyse the statistical behavior of the sparsity-aware SM-NLMS algorithms. Our update equation is given by

$$\mathbf{w}(i+1) = \mathbf{w}(i) + \mu\left(i\right)e\left(i\right)\frac{\mathbf{x}\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2} + \rho\left(i\right)\left(\frac{\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2} - \mathbf{I}_N\right)\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)}. \tag{3-35}$$

Let us define the weight error vector as follows:

$$\mathbf{c}\left(i\right) = \mathbf{w}_o - \mathbf{w}\left(i\right). \tag{3-36}$$

The error from equation (3-31) is now given by the following expression:

$$e\left(i\right) = \mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{c}\left(i\right) + n(i). \tag{3-37}$$

Subtracting $\mathbf{w}\left(i+1\right)$ from $\mathbf{w_o}$ and using equations (3-35), (3-36) and

(3-37), we get

$$\begin{aligned}
\mathbf{c}\left(i+1\right) =& \mathbf{c}\left(i\right) - \frac{\mu\left(i\right)e\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2}\mathbf{x}\left(i\right) \\
& - \rho\left(i\right)\left(\frac{\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2} - \mathbf{I}_N\right)\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)} \\
=& \mathbf{c}\left(i\right) - \frac{\mu\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2}\mathbf{x}\left(i\right)\left(\mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{c}\left(i\right) + n(i)\right) \\
& - \rho\left(i\right)\left(\frac{\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2} - \mathbf{I}_N\right)\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)} \\
=& \left(\mathbf{I}_N - \frac{\mu\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2}\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)\right)\mathbf{c}\left(i\right) - \frac{\mu\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2}\mathbf{x}\left(i\right)n(i) \\
& - \rho\left(i\right)\left(\frac{\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2} - \mathbf{I}_N\right)\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)}. \qquad (3\text{-}38)
\end{aligned}$$

Since the input and the noise are statistically independent, then $\mathbb{E}\left[\mathbf{x}\left(i\right)n(i)\right] = \mathbf{0}$. Taking the expected value from equation (3-38) and assuming that the product $\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)$ is independent from the weights, as described by [54, 58–60], results in

$$\begin{aligned}
\mathbb{E}\left[\mathbf{c}\left(i+1\right)\right] =& \left[\mathbf{I}_N - \mathbb{E}\left[\mu\left(i\right)\right]\mathbb{E}\left[\frac{\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2}\right]\right]\mathbb{E}\left[\mathbf{c}\left(i\right)\right] \\
& - \alpha\mathbb{E}\left[\mu\left(i\right)\right]\left(\mathbb{E}\left[\frac{\mathbf{x}\left(i\right)\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2}\right] - \mathbf{I}_N\right)\mathbb{E}\left[\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)}\right] \\
=& \left[\mathbf{I}_N - \mathbb{E}\left[\mu\left(i\right)\right]\frac{\mathbf{R}_{xx}}{N\sigma_x^2}\right]\mathbb{E}\left[\mathbf{c}\left(i\right)\right] \\
& - \alpha\mathbb{E}\left[\mu\left(i\right)\right]\left(\frac{\mathbf{R}_{xx}}{N\sigma_x^2} - \mathbf{I}_N\right)\mathbb{E}\left[\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)}\right]. \qquad (3\text{-}39)
\end{aligned}$$

As $i \to \infty$, then we have $\mathbb{E}\left[\mathbf{c}\left(i+1\right)\right] \approx \mathbb{E}\left[\mathbf{c}\left(i\right)\right]$ and we obtain

$$\lim_{i\to\infty}\left(\mathbb{E}\left[\mu\left(i\right)\right]\frac{\mathbf{R}_{xx}}{N\sigma_x^2}\mathbb{E}\left[\mathbf{c}\left(i\right)\right]\right) = -\alpha\lim_{i\to\infty}\left(\mathbb{E}\left[\mu\left(i\right)\right]\left(\frac{\mathbf{R}_{xx}}{N\sigma_x^2} - \mathbf{I}_N\right)\mathbb{E}\left[\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)}\right]\right) \qquad (3\text{-}40)$$

Simplifying terms we have

$$\lim_{i\to\infty}\left(\mathbf{w}\left(i\right)\right) - \mathbf{w}_o = \alpha N\sigma_x^2\mathbf{R}_{xx}^{-1}\left(\frac{\mathbf{R}_{xx}}{N\sigma_x^2} - \mathbf{I}_N\right)\mathbb{E}\left[\lim_{i\to\infty}\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)}\right] \qquad (3\text{-}41)$$

$$\lim_{i\to\infty}\left(\mathbf{w}\left(i\right)\right) = \mathbf{w}_o + \alpha\left(\mathbf{I}_N - N\sigma_x^2\mathbf{R}_{xx}^{-1}\right)\left(\frac{\partial f_l\left(\mathbf{w}_o\right)}{\partial \mathbf{w}\left(\mathbf{w}_o\right)}\right). \qquad (3\text{-}42)$$

The last equation indicates that, when applying a penalty function the estimator becomes biased.

3.2.2 Steady-State MSE Analysis

We may represent equation (3-35) as follows:

$$\begin{aligned}\mathbf{w}(i+1) &= \mathbf{w}(i) + P_{up}(i)\left[e(i) - \gamma \mathrm{sgn}(e(i))\right]\frac{\mathbf{x}(i)}{\| \mathbf{x}(i) \|^2} \\ &\quad + P_{up}(i)\,\alpha(i)\left[1 - \frac{\gamma}{|e(i)|}\right]\mathbf{T}(i)\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}, \end{aligned} \tag{3-43}$$

where

$$\mathbf{T}(i) = \left(\frac{\mathbf{x}(i)\,\mathbf{x}^{\mathrm{T}}(i)}{\| \mathbf{x}(i) \|^2} - \mathbf{I}_N\right). \tag{3-44}$$

The variable $P_{up}(i) \in [0,1]$ is a function that represents the probability of updating the filter coefficients at a given iteration i:

$$P_{up}(i) = \mathcal{P}\left[|e(i)| > \gamma\right] \tag{3-45}$$

Let us define the following random variables and matrix to better manipulate the terms:

$$a = \gamma \mathrm{sgn}(e(i)), \tag{3-46}$$

$$b = \frac{\gamma}{|e(i)|}. \tag{3-47}$$

Let $\mathbf{u}(i) \in \mathbb{R}^{N+1}$ be a random vector that represents the update applied to the weights coefficients when $|e(i)| > \gamma$:

$$\begin{aligned}\mathbf{u}(i) &= [e(i) - a]\frac{\mathbf{x}(i)}{\| \mathbf{x}(i) \|^2} \\ &\quad + \alpha(i)\,[1-b]\,\mathbf{T}(i)\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}. \end{aligned} \tag{3-48}$$

The update equation is now given by

$$\mathbf{w}(i+1) = \begin{cases} \mathbf{w}(i) + \mathbf{u}(i) & |e(i)| > \gamma, \\ \mathbf{w}(i) & \text{otherwise.} \end{cases} \tag{3-49}$$

Consider the vector $\mathbf{u}'(i)$ defined by

$$\mathbf{u}'(i) = \begin{cases} \mathbf{u}(i) & |e(i)| > \gamma \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{3-50}$$

Let us now rewrite the update equation and take the expected value as

follows:

$$\begin{aligned} \mathbf{w}(i+1) &= \mathbf{w}(i) + \mathbf{u}'(i), && \text{(3-51)} \\ \mathbb{E}\left[\mathbf{w}(i+1)\right] &= \mathbb{E}\left[\mathbf{w}(i)\right] + \mathbb{E}\left[\mathbf{u}'(i)\right]. && \text{(3-52)} \end{aligned}$$

Using the total probability theorem to evaluate $\mathbb{E}\left[\mathbf{u}'(i)\right]$ we arrive at:

$$\begin{aligned} \mathbb{E}\left[\mathbf{u}'(i)\right] = &\ \mathbb{E}\left[\mathbf{u}'(i) | \left\{|e\left(i\right)| \leq \gamma\right\}\right]\left(1 - P_{up}\left(i\right)\right) \\ &+ \mathbb{E}\left[\mathbf{u}'(i) | \left\{|e\left(i\right)| > \gamma\right\}\right] P_{up}\left(i\right), \\ = &\ 0 + P_{up}\left(i\right)\mathbb{E}\left[\mathbf{u}(i)\right]. \end{aligned} \quad \text{(3-53)}$$

Taking the expected value from equation (3-43), we obtain

$$\begin{aligned} \mathbb{E}\left[\mathbf{w}(i+1)\right] &= \mathbb{E}\left[\mathbf{w}(i)\right] + P_{up}\left(i\right)\mathbb{E}\left[\mathbf{u}(i)\right], \\ &= \mathbb{E}\left[\mathbf{w}(i)\right] + \mathbb{E}\left[\mathbf{u}'(i)\right], \end{aligned} \quad \text{(3-54)}$$

which proves that equation (3-35) and (3-43) are equal at least in the mean. Transposing and post-multiplying equation (3-43) by $\mathbf{x}\left(i\right)$ results in

$$\begin{aligned} \mathbf{w}^{\mathrm{T}}(i+1)\mathbf{x}\left(i\right) &= \mathbf{w}^{\mathrm{T}}(i)\mathbf{x}\left(i\right) + P_{up}\left(i\right)\left[e\left(i\right) - a\right]\frac{\mathbf{x}^{\mathrm{T}}\left(i\right)}{\parallel \mathbf{x}\left(i\right) \parallel^2}\mathbf{x}\left(i\right) \\ &\quad + P_{up}\left(i\right)\alpha\left(i\right)\left[1 - b\right]\frac{\partial f_l\left(\mathbf{w}^{\mathrm{T}}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)}\mathbf{T}\left(i\right)\mathbf{x}\left(i\right), \\ &= \mathbf{w}^{\mathrm{T}}(i)\mathbf{x}\left(i\right) + P_{up}\left(i\right)\left[e\left(i\right) - a\right] \\ &\quad + P_{up}\left(i\right)\alpha\left(i\right)\left[1 - b\right]\frac{\partial f_l\left(\mathbf{w}^{\mathrm{T}}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)}\left(\mathbf{x}\left(i\right) - \mathbf{x}\left(i\right)\right), \\ &= \mathbf{w}^{\mathrm{T}}(i)\mathbf{x}\left(i\right) + P_{up}\left(i\right)\left[e\left(i\right) - a\right]. \end{aligned} \quad \text{(3-55)}$$

Substituting the last equation in (3-33), we get

$$\begin{aligned} e_p\left(i\right) &= \mathbf{w_o}^{\mathrm{T}}\mathbf{x}\left(i\right) - \mathbf{w}^{\mathrm{T}}(i)\mathbf{x}\left(i\right) - P_{up}\left(i\right)\left[e\left(i\right) - a\right] \\ &= e_a\left(i\right) - P_{up}\left(i\right)\left[e\left(i\right) - a\right]. \end{aligned} \quad \text{(3-56)}$$

The *a posteriori* error can be expressed in terms of the error signal as follows:

$$\begin{aligned} e_p\left(i\right) &= e\left(i\right) - n\left(i\right) - P_{up}\left(i\right)\left[e\left(i\right) - a\right] \\ &= \left(1 - P_{up}\left(i\right)\right)e\left(i\right) + P_{up}\left(i\right)a - n\left(i\right). \end{aligned} \quad \text{(3-57)}$$

Solving equation (3-56) for $e\left(i\right)$, we obtain

$$e\left(i\right)=\frac{1}{P_{up}\left(i\right)}\left(e_{a}\left(i\right)-e_{p}\left(i\right)+P_{up}\left(i\right)a\right). \tag{3-58}$$

Substituting (3-58) in (3-43), yields

$$\begin{aligned}\mathbf{w}(i+1) &= \mathbf{w}(i)+\frac{P_{up}\left(i\right)}{P_{up}\left(i\right)}\left[e_{a}\left(i\right)-e_{p}\left(i\right)+P_{up}\left(i\right)a-P_{up}\left(i\right)a\right]\frac{\mathbf{x}\left(i\right)}{\parallel\mathbf{x}\left(i\right)\parallel^{2}}\\ &\quad+P_{up}\left(i\right)\alpha\left(i\right)\left[1-b\right]\mathbf{T}\left(i\right)\frac{\partial f_{l}\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)},\\ &= \mathbf{w}(i)+\left[e_{a}\left(i\right)-e_{p}\left(i\right)\right]\frac{\mathbf{x}\left(i\right)}{\parallel\mathbf{x}\left(i\right)\parallel^{2}}\\ &\quad+P_{up}\left(i\right)\alpha\left(i\right)\left[1-b\right]\mathbf{T}\left(i\right)\frac{\partial f_{l}\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}.\end{aligned} \tag{3-59}$$

Subtracting $\mathbf{w}_{o}-\mathbf{w}\left(i+1\right)$,we have

$$\begin{aligned}\mathbf{c}\left(i+1\right)+e_{a}\left(i\right)\frac{\mathbf{x}\left(i\right)}{\parallel\mathbf{x}\left(i\right)\parallel^{2}} =&\mathbf{c}(i)+e_{p}\left(i\right)\frac{\mathbf{x}\left(i\right)}{\parallel\mathbf{x}\left(i\right)\parallel^{2}}\\ &-P_{up}\left(i\right)\alpha\left(i\right)\left[1-b\right]\mathbf{T}\left(i\right)\frac{\partial f_{l}\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}.\end{aligned} \tag{3-60}$$

Pre-multiplying equation (3-60) by its transpose as described in [61], we arrive at

$$\begin{aligned}LHS =&\left\|\mathbf{c}\left(i+1\right)\right\|^{2}+\mathbf{c}^{\mathrm{T}}\left(i+1\right)\mathbf{x}\left(i\right)\frac{e_{a}\left(i\right)}{\left\|\mathbf{x}\left(i\right)\right\|^{2}}\\ &+\frac{e_{a}\left(i\right)}{\left\|\mathbf{x}\left(i\right)\right\|^{2}}\mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{c}\left(i+1\right)+\frac{e_{a}^{2}\left(i\right)}{\left\|\mathbf{x}\left(i\right)\right\|^{2}},\end{aligned} \tag{3-61}$$

$$\begin{aligned}RHS =&\left\|\mathbf{c}\left(i\right)\right\|^{2}+\mathbf{c}^{\mathrm{T}}\left(i\right)\mathbf{x}\left(i\right)\frac{e_{p}\left(i\right)}{\left\|\mathbf{x}\left(i\right)\right\|^{2}}+\frac{e_{p}\left(i\right)}{\left\|\mathbf{x}\left(i\right)\right\|^{2}}\mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{c}\left(i\right)\\ &-P_{up}\left(i\right)\alpha\left(i\right)\left[1-b\right]\mathbf{c}^{\mathrm{T}}\left(i\right)\mathbf{T}\left(i\right)\frac{\partial f_{l}\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}+\frac{e_{p}^{2}\left(i\right)}{\left\|\mathbf{x}\left(i\right)\right\|^{2}}\\ &-P_{up}\left(i\right)\alpha\left(i\right)\left[1-b\right]\frac{e_{p}\left(i\right)}{\left\|\mathbf{x}\left(i\right)\right\|^{2}}\mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{T}\left(i\right)\frac{\partial f_{l}\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}\\ &-P_{up}\left(i\right)\alpha\left(i\right)\left[1-b\right]\frac{\partial f_{l}\left(\mathbf{w}^{\mathrm{T}}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}\mathbf{T}\left(i\right)\mathbf{c}\left(i\right)\\ &-P_{up}\left(i\right)\alpha\left(i\right)\left[1-b\right]\frac{\partial f_{l}\left(\mathbf{w}^{\mathrm{T}}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}\mathbf{T}\left(i\right)\mathbf{x}\left(i\right)\frac{e_{p}\left(i\right)}{\left\|\mathbf{x}\left(i\right)\right\|^{2}}\\ &+\left(P_{up}\left(i\right)\alpha\left(i\right)\left[1-b\right]\right)^{2}\frac{\partial f_{l}\left(\mathbf{w}^{\mathrm{T}}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}\mathbf{T}^{\mathrm{T}}\left(i\right)\mathbf{T}\left(i\right)\frac{\partial f_{l}\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}.\end{aligned} \tag{3-62}$$

The product

$$\mathbf{x}^{\mathrm{T}}\left(i\right)\mathbf{T}\left(i\right)=\mathbf{x}^{\mathrm{T}}\left(i\right)-\mathbf{x}^{\mathrm{T}}\left(i\right)=0, \tag{3-63}$$

which means that the sixth and the eighth terms go to zero. We also have that $\mathbf{T}^{\mathrm{T}}(i)\,\mathbf{T}(i) = -\mathbf{T}(i)$. Replacing equation (3-32) in (3-62) and equation (3-33) in (3-61), we get

$$\begin{aligned} LHS = & \left\|\mathbf{c}(i+1)\right\|^2 + e_p(i)\,\frac{e_a(i)}{\left\|\mathbf{x}(i)\right\|^2} \\ & + \frac{e_a(i)}{\left\|\mathbf{x}(i)\right\|^2} e_p(i) + \frac{e_a^2(i)}{\left\|\mathbf{x}(i)\right\|^2}, \end{aligned} \tag{3-64}$$

$$\begin{aligned} RHS = & \left\|\mathbf{c}(i)\right\|^2 + e_a(i)\,\frac{e_p(i)}{\left\|\mathbf{x}(i)\right\|^2} + \frac{e_p(i)}{\left\|\mathbf{x}(i)\right\|^2} e_a(i) \\ & - P_{up}(i)\,\alpha(i)\,[1-b]\,\mathbf{c}^{\mathrm{T}}(i)\,\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)} + \frac{e_p^2(i)}{\left\|\mathbf{x}(i)\right\|^2} \\ & - P_{up}(i)\,\alpha(i)\,[1-b]\,\frac{\partial f_l\left(\mathbf{w}^{\mathrm{T}}(i)\right)}{\partial \mathbf{w}(i)}\mathbf{T}(i)\,\mathbf{c}(i) \\ & - \left(P_{up}(i)\,\alpha(i)\,[1-b]\right)^2 \frac{\partial f_l\left(\mathbf{w}^{\mathrm{T}}(i)\right)}{\partial \mathbf{w}(i)}\left(\mathbf{T}(i)\right)\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}. \end{aligned} \tag{3-65}$$

Eliminating the terms that are equal on both sides, we obtain

$$\frac{e_a^2(i)}{\left\|\mathbf{x}(i)\right\|^2} = \left\|\mathbf{c}(i)\right\|^2 - \left\|\mathbf{c}(i+1)\right\|^2 + \frac{e_p^2(i)}{\left\|\mathbf{x}(i)\right\|^2} - P_{up}(i)\,s(i), \tag{3-66}$$

where the variable $s(i)$ is defined as follows:

$$\begin{aligned} s(i) = & \alpha(i)\,[1-b]\,\mathbf{c}^{\mathrm{T}}(i)\,\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)} \\ & + \alpha(i)\,[1-b]\,\frac{\partial f_l\left(\mathbf{w}^{\mathrm{T}}(i)\right)}{\partial \mathbf{w}(i)}\mathbf{T}(i)\,\mathbf{c}(i) \\ & + P_{up}(i)\left(\alpha(i)\,[1-b]\right)^2 \frac{\partial f_l\left(\mathbf{w}^{\mathrm{T}}(i)\right)}{\partial \mathbf{w}(i)}\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)} \end{aligned} \tag{3-67}$$

Taking the expected value from (3-66) leads to:

$$\mathbb{E}\left[\frac{e_a^2(i)}{\left\|\mathbf{x}(i)\right\|^2}\right] = \mathbb{E}\left[\left\|\mathbf{c}(i)\right\|^2\right] - \mathbb{E}\left[\left\|\mathbf{c}(i+1)\right\|^2\right] + \mathbb{E}\left[\frac{e_p^2(i)}{\left\|\mathbf{x}(i)\right\|^2}\right] - P_{up}(i)\,\mathbb{E}\left[s(i)\right] \tag{3-68}$$

As $i \to \infty$ the probability of update becomes a constant and, as mentioned in [57, 62, 63], it is given by

$$P_{up} = 2Q\left(\frac{\gamma}{\sigma_e}\right). \tag{3-69}$$

Another important consequence is that as $i \to \infty$, then $\mathbb{E}\left[\left\|\mathbf{c}(i)\right\|^2\right] \approx \mathbb{E}\left[\left\|\mathbf{c}(i+1)\right\|^2\right]$. Simplifying the terms of equation (3-68) results in

$$\lim_{i\to\infty}\left(\mathbb{E}\left[\frac{e_a^2(i)}{\|\mathbf{x}(i)\|^2}\right]\right) = \lim_{i\to\infty}\left(\mathbb{E}\left[\frac{e_p^2(i)}{\|\mathbf{x}(i)\|^2}\right]\right) - P_{up}\lim_{i\to\infty}\left(\mathbb{E}\left[s(i)\right]\right), \quad (3\text{-}70)$$

$$P_{up}\lim_{i\to\infty}\left(\mathbb{E}\left[s(i)\right]\right) = \lim_{i\to\infty}\left(\mathbb{E}\left[\frac{e_p^2(i)}{\|\mathbf{x}(i)\|^2}\right] - \mathbb{E}\left[\frac{e_a^2(i)}{\|\mathbf{x}(i)\|^2}\right]\right), \quad (3\text{-}71)$$

Let us now use the following results [54]:

$$\mathbb{E}\left[\frac{e_a^2(i)}{\|\mathbf{x}(i)\|^2}\right] \approx \frac{\mathbb{E}\left[e_a^2(i)\right]}{N\sigma_x^2}, \quad (3\text{-}72)$$

$$\mathbb{E}\left[\frac{e_p^2(i)}{\|\mathbf{x}(i)\|^2}\right] \approx \frac{\mathbb{E}\left[e_p^2(i)\right]}{N\sigma_x^2}, \quad (3\text{-}73)$$

and substitute them into equation (3-71), leading us to

$$P_{up}\lim_{i\to\infty}\left(\mathbb{E}\left[s(i)\right]\right) = \lim_{i\to\infty}\left(\frac{\mathbb{E}\left[e_p^2(i)\right] - \mathbb{E}\left[e_a^2(i)\right]}{N\sigma_x^2}\right), \quad (3\text{-}74)$$

$$P_{up}N\sigma_x^2\lim_{i\to\infty}\left(\mathbb{E}\left[s(i)\right]\right) = \lim_{i\to\infty}\left(\mathbb{E}\left[e_p^2(i)\right] - \mathbb{E}\left[e_a^2(i)\right]\right). \quad (3\text{-}75)$$

Using equation (3-34) to evaluate the term $\mathbb{E}\left[e_a^2(i)\right]$ we arrive at the following expression:

$$\begin{aligned}
\mathbb{E}\left[e_a^2(i)\right] &= \mathbb{E}\left[\left(e(i) - n(i)\right)^2\right] \\
&= \mathbb{E}\left[e^2(i)\right] - 2\mathbb{E}\left[e(i)\,n(i)\right] + \mathbb{E}\left[n^2(i)\right] \\
&= \mathbb{E}\left[e^2(i)\right] - 2\mathbb{E}\left[\left(\mathbf{c}^{\mathrm{T}}(i)\,\mathbf{x}(i) + n(i)\right)n(i)\right] + \sigma_n^2 \\
&= \mathbb{E}\left[e^2(i)\right] - 2\sigma_n^2 + \sigma_n^2 \\
&= \mathbb{E}\left[e^2(i)\right] - \sigma_n^2
\end{aligned} \quad (3\text{-}76)$$

Let us now evaluate the term involving the *a posteriori* error using equation (3-57):

$$\begin{aligned}
\mathbb{E}\left[e_p^2(i)\right] &= \mathbb{E}\left[\left(\left(1 - P_{up}\right)e(i) + P_{up}a - n(i)\right)^2\right] \\
&= \left(1 - P_{up}\right)^2\mathbb{E}\left[e^2(i)\right] - 2\left(1 - P_{up}\right)\mathbb{E}\left[e(i)\,n(i)\right] - 2P_{up}\mathbb{E}\left[an(i)\right] \\
&\quad +2\left(1 - P_{up}\right)P_{up}\mathbb{E}\left[e(i)\,a\right] + P_{up}^2\mathbb{E}\left[a^2\right] + \mathbb{E}\left[n^2(i)\right] \\
&= \left(1 - P_{up}\right)^2\mathbb{E}\left[e^2(i)\right] - 2\left(1 - P_{up}\right)\sigma_n^2 - 2P_{up}\gamma\mathbb{E}\left[\mathrm{sgn}\left(e(i)\right)n(i)\right] \\
&\quad +2\left(1 - P_{up}\right)P_{up}\gamma\mathbb{E}\left[\mathrm{sgn}\left(e(i)\right)e(i)\right] + P_{up}^2\gamma^2 + \sigma_n^2
\end{aligned} \quad (3\text{-}77)$$

Using Price's theorem [63, 64], we obtain

$$\mathbb{E}\left[\operatorname{sgn}\left(q\right)\mathbf{p}\right] \approx \sqrt{\frac{2}{\pi\sigma_q}}\mathbb{E}\left[q\mathbf{p}\right], \tag{3-78}$$

where q and $\mathbf{p}$ are a random variable and a random vector respectively. Applying this relation on equation (3-77), we get

$$\begin{aligned}
\mathbb{E}\left[e_p^2\left(i\right)\right] &= \left(1-P_{up}\right)^2\mathbb{E}\left[e^2\left(i\right)\right] + 2\left(1-P_{up}\right)P_{up}\gamma\sqrt{\frac{2}{\pi\sigma_e}}\mathbb{E}\left[e^2\left(i\right)\right] \\
&\quad -2P_{up}\gamma\sqrt{\frac{2}{\pi\sigma_e}}\mathbb{E}\left[e\left(i\right)n\left(i\right)\right] + P_{up}^2\gamma^2 - \left(1-2P_{up}\right)\sigma_n^2 \\
&= \left(1-P_{up}\right)^2\mathbb{E}\left[e^2\left(i\right)\right] + 2\left(1-P_{up}\right)P_{up}\gamma\sqrt{\frac{2}{\pi\sigma_e}}\mathbb{E}\left[e^2\left(i\right)\right] \\
&\quad -2P_{up}\gamma\sqrt{\frac{2}{\pi\sigma_e}}\sigma_n^2 + P_{up}^2\gamma^2 - \left(1-2P_{up}\right)\sigma_n^2. \qquad (3\text{-}79)
\end{aligned}$$

Subtracting (3-76) and (3-79) yields

$$\begin{aligned}
\mathbb{E}\left[e_p^2\left(i\right)\right] - \mathbb{E}\left[e_a^2\left(i\right)\right] &= \left(1-P_{up}\right)^2\mathbb{E}\left[e^2\left(i\right)\right] \\
&\quad +2\left(1-P_{up}\right)P_{up}\gamma\sqrt{\frac{2}{\pi\sigma_e}}\mathbb{E}\left[e^2\left(i\right)\right] \\
&\quad -2P_{up}\gamma\sqrt{\frac{2}{\pi\sigma_e}}\sigma_n^2 + P_{up}^2\gamma^2 - \left(1-2P_{up}\right)\sigma_n^2 \\
&\quad -\mathbb{E}\left[e^2\left(i\right)\right] + \sigma_n^2 \\
&= \left(P_{up}^2 - 2P_{up}\right)\mathbb{E}\left[e^2\left(i\right)\right] \\
&\quad +2\left(1-P_{up}\right)P_{up}\gamma\sqrt{\frac{2}{\pi\sigma_e}}\mathbb{E}\left[e^2\left(i\right)\right] \\
&\quad -2P_{up}\gamma\sqrt{\frac{2}{\pi\sigma_e}}\sigma_n^2 + P_{up}^2\gamma^2 + 2P_{up}\sigma_n^2 \\
&= P_{up}\left(P_{up} - 2 + 2\left(1-P_{up}\right)\gamma\rho_e\right)\mathbb{E}\left[e^2\left(i\right)\right] \\
&\quad +2P_{up}\left(1-\gamma\rho_e\right)\sigma_n^2 + P_{up}^2\gamma^2, \qquad (3\text{-}80)
\end{aligned}$$

where

$$\rho_e = \sqrt{\frac{2}{\pi\sigma_e}}. \tag{3-81}$$

Using equation (3-80) in (3-75), we have

$$\begin{aligned}
P_{up}N\sigma_x^2\lim_{i\to\infty}\left(\mathbb{E}\left[s\left(i\right)\right]\right) &= P_{up}\left(P_{up} - 2 + 2\left(1-P_{up}\right)\gamma\rho_e\right)\lim_{i\to\infty}\mathbb{E}\left[e^2\left(i\right)\right] \\
&\quad +2P_{up}\left(1-\gamma\rho_e\right)\sigma_n^2 + P_{up}^2\gamma^2, \qquad (3\text{-}82) \\
N\sigma_x^2\lim_{i\to\infty}\left(\mathbb{E}\left[s\left(i\right)\right]\right) &= \left(P_{up} - 2 + 2\left(1-P_{up}\right)\gamma\rho_e\right)\lim_{i\to\infty}\mathbb{E}\left[e^2\left(i\right)\right] \\
&\quad +2\left(1-\gamma\rho_e\right)\sigma_n^2 + P_{up}\gamma^2, \qquad (3\text{-}83)
\end{aligned}$$

By isolating the term $\lim_{i\to\infty}\mathbb{E}\left[e^2\left(i\right)\right]=MSE$, we obtain

$$\left(2-P_{up}-2\left(1-P_{up}\right)\gamma\rho_e\right)MSE=2\left(1-\gamma\rho_e\right)\sigma_n^2+P_{up}\gamma^2-N\sigma_x^2\lim_{i\to\infty}\left(\mathbb{E}\left[s\left(i\right)\right]\right). \tag{3-84}$$

Let us denote the MSE for the sparsity-aware SM-NLMS algorithms with an arbitrary penalty function by MSE_{SA-SM}. Solving equation (3-84) for MSE_{SA-SM}, we get

$$\begin{aligned} MSE_{SA-SM} &= \frac{2\left(1-\gamma\rho_e\right)\sigma_n^2+P_{up}\gamma^2}{\left(2-P_{up}-2\left(1-P_{up}\right)\gamma\rho_e\right)} \\ &\quad -\frac{N\sigma_x^2}{\left(2-P_{up}-2\left(1-P_{up}\right)\gamma\rho_e\right)}\lim_{i\to\infty}\left(\mathbb{E}\left[s\left(i\right)\right]\right). \end{aligned} \tag{3-85}$$

In [62], [57] and [63] the EMSE of the SM-NLMS algorithm is presented. From this result the MSE can be calculated and is given by

$$MSE_{SM-NLMS}=\frac{2\left(1-\gamma\rho_e\right)\sigma_n^2+P_{up}\gamma^2}{\left(2-P_{up}-2\left(1-P_{up}\right)\gamma\rho_e\right)} \tag{3-86}$$

This means that equation (3-85) becomes

$$MSE_{SA-SM}=MSE_{SM-NLMS}-\frac{N\sigma_x^2}{\lambda}\lim_{i\to\infty}\left(\mathbb{E}\left[s\left(i\right)\right]\right), \tag{3-87}$$

where

$$\lambda=\left(2-P_{up}-2\left(1-P_{up}\right)\gamma\rho_e\right). \tag{3-88}$$

Let us now focus on the value of $\mathbb{E}\left[s\left(i\right)\right]$ and assuming that the parameter α is a fixed scalar, we obtain

$$\begin{aligned} \mathbb{E}\left[s\left(i\right)\right] &= \alpha\mathbb{E}\left[\left[1-b\right]\mathbf{c}^{\mathrm{T}}\left(i\right)\mathbf{T}\left(i\right)\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}\right] \\ &\quad +\alpha\mathbb{E}\left[\left[1-b\right]\frac{\partial f_l\left(\mathbf{w}^{\mathrm{T}}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}\mathbf{T}\left(i\right)\mathbf{c}\left(i\right)\right] \\ &\quad +P_{up}\alpha^2\mathbb{E}\left[\left[1-b\right]^2\frac{\partial f_l\left(\mathbf{w}^{\mathrm{T}}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}\mathbf{T}\left(i\right)\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial\mathbf{w}\left(i\right)}\right]. \end{aligned} \tag{3-89}$$

Usually α is set up in the range of $[10^{-4},10^{-3}]$ because the best performance of the algorithms is achieved by this range. A higher value of α will deteriorate the performance of the algorithms. The range of P_{up} determined experimentally for several γ at steady-state is $[10^{-2},10^{-1}]$. This means that $P_{up}\alpha^2$ is in the range from 10^{-10} to 10^{-7} and we can neglect the last term. Since $\mathbf{T}\left(i\right)$ is a symmetric matrix, then $\mathbf{c}^{\mathrm{T}}\left(i\right)\mathbf{T}\left(i\right)\frac{\partial f_l(\mathbf{w}(i))}{\partial\mathbf{w}(i)}\triangleq\frac{\partial f_l\left(\mathbf{w}^{\mathrm{T}}(i)\right)}{\partial\mathbf{w}(i)}\mathbf{T}\left(i\right)\mathbf{c}\left(i\right)$. With

these results, the value of $\mathbb{E}[s(i)]$ is reduced to:

$$\begin{aligned}\mathbb{E}[s(i)] =& 2\alpha\mathbb{E}\left[[1-b]\,\mathbf{c}^{\mathrm{T}}(i)\,\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right] \\ =& 2\alpha\mathbb{E}\left[\mathbf{c}^{\mathrm{T}}(i)\,\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right] \\ & - 2\alpha\gamma\mathbb{E}\left[\frac{a}{e(i)}\mathbf{c}^{\mathrm{T}}(i)\,\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right] \\ =& 2\alpha\mathbb{E}\left[\mathbf{c}^{\mathrm{T}}(i)\,\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right] \\ & - 2\alpha\gamma\mathbb{E}\left[\frac{\mathrm{sgn}(e(i))}{e(i)}\mathbf{c}^{\mathrm{T}}(i)\,\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]. \end{aligned} \tag{3-90}$$

Applying once again Price's theorem over the last equation leads to:

$$\begin{aligned}\mathbb{E}[s(i)] =& 2\alpha(1-\gamma\rho_e)\,\mathbb{E}\left[\mathbf{c}^{\mathrm{T}}(i)\,\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right] \\ =& 2\alpha(1-\gamma\rho_e)\,\mathbf{w_o}^{\mathrm{T}}\mathbb{E}\left[\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right] \\ & - 2\alpha(1-\gamma\rho_e)\,\mathbb{E}\left[\mathbf{w}^{\mathrm{T}}(i)\,\mathbf{T}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]. \end{aligned} \tag{3-91}$$

Replacing (3-44) on the last equation, we have

$$\begin{aligned}\mathbb{E}[s(i)] =& 2\alpha(1-\gamma\rho_e)\,\mathbf{w_o}^{\mathrm{T}}\mathbb{E}\left[\frac{\mathbf{x}(i)\,\mathbf{x}^{\mathrm{T}}(i)}{\|\,\mathbf{x}(i)\,\|^2}\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right] \\ & -2\alpha(1-\gamma\rho_e)\,\mathbf{w_o}^{\mathrm{T}}\mathbb{E}\left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right] \\ & -2\alpha(1-\gamma\rho_e)\,\mathbb{E}\left[\mathrm{tr}\left\{\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\mathbf{w}^{\mathrm{T}}(i)\,\frac{\mathbf{x}(i)\,\mathbf{x}^{\mathrm{T}}(i)}{\|\,\mathbf{x}(i)\,\|^2}\right\}\right] \\ & +2\alpha(1-\gamma\rho_e)\,\mathbb{E}\left[\mathbf{w}^{\mathrm{T}}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right] \\ =& 2\alpha(1-\gamma\rho_e)\left\{\mathbf{w_o}^{\mathrm{T}}\left(\frac{\mathbf{R}_{xx}}{N\sigma_x^2}-\mathbf{I}_N\right)\mathbb{E}\left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]\right\} \\ & -2\alpha(1-\gamma\rho_e)\,\mathrm{tr}\left\{\mathbb{E}\left[\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\mathbf{w}^{\mathrm{T}}(i)\right]\frac{\mathbf{R}_{xx}}{N\sigma_x^2}\right\} \\ & +2\alpha(1-\gamma\rho_e)\,\mathbb{E}\left[\mathbf{w}^{\mathrm{T}}(i)\,\frac{\partial f_l(\mathbf{w}(i))}{\partial \mathbf{w}(i)}\right]. \end{aligned} \tag{3-92}$$

The final expression for the MSE_{SA-SM} is given by

$$\begin{aligned} MSE_{SA-SM} &= MSE_{SM-NLMS} - \beta \left\{ \mathbf{w_o}^{\mathrm{T}} \left(\frac{\mathbf{R}_{xx}}{N\sigma_x^2} - \mathbf{I}_N \right) \mathbb{E}\left[\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)} \right] \right\} \\ &+ \beta \mathrm{tr} \left\{ \mathbb{E}\left[\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)} \mathbf{w}^{\mathrm{T}}\left(i\right) \right] \frac{\mathbf{R}_{xx}}{N\sigma_x^2} \right\} \\ &- \beta \mathbb{E}\left[\mathbf{w}^{\mathrm{T}}\left(i\right) \frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)} \right], \end{aligned} \tag{3-93}$$

where

$$\beta = \frac{N\sigma_x^2 2\alpha\left(1-\gamma\rho_e\right)}{\lambda}. \tag{3-94}$$

3.2.3
The steady-state MSE of the ZA-SM-NLMS algorithm

Let us consider the zero attracting approach, that employs the $l_1 - norm$ as the penalty function:

$$\frac{\partial f_l\left(\mathbf{w}\left(i\right)\right)}{\partial \mathbf{w}\left(i\right)} = \mathrm{sgn}(\mathbf{w}\left(i\right)) \tag{3-95}$$

Assuming that $\mathrm{w}_j\left(i\right)$ has a Gaussian distribution $\mathcal{N}(m_w, \sigma_w^2)$, we may evaluate every entry of vector $\mathbb{E}\left[\mathrm{sgn}(\mathbf{w}\left(i\right))\right]$ as sugested in [65].

$$\begin{aligned} \mathbb{E}\left[sgn(\mathrm{w}\left(i\right))\right] &= \int_{-\infty}^{\infty} sgn(\mathrm{W}\left(i\right)) f_{\mathrm{w}}\left\{\mathrm{W}\left(i\right)\right\} d\mathrm{W}\left(i\right) \\ &= \int_{-\infty}^{0} -\frac{1}{\sigma_w\sqrt{2\pi}} e^{-\frac{\left(\mathrm{W}-m_w\right)^2}{2\sigma_w^2}} d\mathrm{W}\left(i\right) \\ &\quad + \int_{0}^{\infty} \frac{1}{\sigma_w\sqrt{2\pi}} e^{-\frac{\left(\mathrm{W}-m_w\right)^2}{2\sigma_w^2}} d\mathrm{W}\left(i\right) \\ &= -1 + 2\int_{0}^{\infty} \frac{1}{\sigma_w\sqrt{2\pi}} e^{-\frac{\left(\mathrm{W}-m_w\right)^2}{2\sigma_w^2}} d\mathrm{W}\left(i\right). \end{aligned} \tag{3-96}$$

By substituting the variable $z = \frac{\left(\mathrm{W}-m_w\right)}{\sigma_w}$, we obtain that each entry of vector $\mathbb{E}\left[\mathrm{sgn}(\mathbf{w}\left(i\right))\right]$ is given by

$$\begin{aligned} \mathbb{E}\left[sgn(\mathrm{w}\left(i\right))\right] &= -1 + 2\int_{-\frac{m_w}{\sigma_w}}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}} dz \\ &= -1 + 2Q\left(-\frac{m_w}{\sigma_w}\right) \end{aligned} \tag{3-97}$$

Let us now consider a white input. The elements of the main diagonal of the third term of equation (3-93) requires the following computation:

$$\begin{aligned}\mathbb{E}\left[\mathrm{w}\left(i\right)sgn(\mathrm{w}\left(i\right))\right] &= \int_{-\infty}^{\infty}\mathrm{w}\left(i\right)sgn(\mathrm{W}\left(i\right))f_{\mathrm{w}}\left\{\mathrm{W}\left(i\right)\right\}d\mathrm{W}\left(i\right)\\ &= \int_{-\infty}^{0}-\mathrm{w}\frac{1}{\sigma_w\sqrt{2\pi}}e^{-\frac{(\mathrm{W}-m_w)^2}{2\sigma_w^2}}d\mathrm{W}\left(i\right)\\ &\quad+\int_{0}^{\infty}\mathrm{w}\frac{1}{\sigma_w\sqrt{2\pi}}e^{-\frac{(\mathrm{W}-m_w)^2}{2\sigma_w^2}}d\mathrm{W}\left(i\right)\\ &= -1+2\int_{0}^{\infty}\mathrm{w}\frac{1}{\sigma_w\sqrt{2\pi}}e^{-\frac{(\mathrm{W}-m_w)^2}{2\sigma_w^2}}d\mathrm{W}\left(i\right). \quad (3\text{-}98)\end{aligned}$$

Replacing by $\mathrm{w}=z\sigma_w+m_w$ leads to:

$$\begin{aligned}\mathbb{E}\left[\mathrm{w}\left(i\right)sgn(\mathrm{w}\left(i\right))\right] &= -1+2\int_{-\frac{m_w}{\sigma_w}}^{\infty}\left(z\sigma_w+m_w\right)\frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2}}dz\\ &= 1+2\sigma_w\int_{-\frac{m_w}{\sigma_w}}^{\infty}z\frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2}}dz\\ &\quad+2m_w\int_{-\frac{m_w}{\sigma_w}}^{\infty}\frac{1}{\sqrt{2\pi}}e^{-\frac{z^2}{2}}dz. \quad (3\text{-}99)\end{aligned}$$

Let us now define $u=-z^2/2$ and $du=-zdz$ to solve the integral. The main diagonal of matrix $\mathbb{E}\left[\mathrm{sgn}(\mathbf{w}\left(i\right))\mathbf{w}^{\mathrm{T}}\left(i\right)\right]$ can be calculated by

$$\begin{aligned}\mathbb{E}\left[\mathrm{w}\left(i\right)sgn(\mathrm{w}\left(i\right))\right] &= -1+2\sigma_w\int_{-\frac{m_w}{\sigma_w}}^{\infty}\frac{1}{\sqrt{2\pi}}e^{u}du+2m_wQ\left(-\frac{m_w}{\sigma_w}\right)\\ &= -1+\frac{2\sigma_w}{\sqrt{2\pi}}e^{-\frac{m_w^2}{2\sigma_w^2}}+2m_wQ\left(-\frac{m_w}{\sigma_w}\right). \quad (3\text{-}100)\end{aligned}$$

Substituting (3-97) and (3-100) in (3-93) leads to the mathematical expression of the MSE for the ZA-SM-NLMS algorithm, which is given by

$$MSE_{ZA-SM} = MSE_{SM-NLMS}-\frac{N\sigma_x^2 2\alpha\left(1-\gamma\rho_e\right)}{\lambda}g, \quad (3\text{-}101)$$

where

$$\begin{aligned}g =&\mathbf{w_o}^{\mathrm{T}}\left(\frac{\mathbf{R}_{xx}}{N\sigma_x^2}-\mathbf{I}_N\right)\mathbf{h}-\mathrm{tr}\left\{\frac{\mathbf{G}\mathbf{R}_{xx}}{N\sigma_x^2}\right\}\\ &+\sum_{i=1}^{N}\left[\frac{2\sigma_{w_i}}{\sqrt{2\pi}}e^{-\frac{m_{w_i}^2}{2\sigma_{w_i}^2}}+2m_{w_i}Q\left(-\frac{m_{w_i}}{\sigma_{w_i}}\right)-1\right], \quad (3\text{-}102)\end{aligned}$$

$$[\mathbf{G}]_{ii}=\frac{2\sigma_{w_i}}{\sqrt{2\pi}}e^{-\frac{m_{w_i}^2}{2\sigma_{w_i}^2}}+2m_{w_i}Q\left(-\frac{m_{w_i}}{\sigma_{w_i}}\right)-1, \quad (3\text{-}103)$$

$$\mathbf{h} = \begin{bmatrix} 2Q\left(-\frac{m_{w_1}}{\sigma_{w_1}}\right) - 1 \\ 2Q\left(-\frac{m_{w_2}}{\sigma_{w_2}}\right) - 1 \\ \vdots \\ 2Q\left(-\frac{m_{w_N}}{\sigma_{w_N}}\right) - 1 \end{bmatrix}. \tag{3-104}$$

3.3 Simulations

In this section several experiments were carried out to test the proposed algorithms and compare them with conventional sparsity-aware adaptive algorithms. For this purpose, a system identification task was chosen. The mathematical formulation of the steady-state MSE obtained in the previous section is also analysed.

3.3.1 Learning Performance of Proposed and Existing Algorithms

To evaluate the algorithms developed, many experiments were set up. For this purpose we consider a system modelled as a finite impulse response (FIR) filter with 64 taps in three different scenarios. The first scenario represents a sparse system where only four taps have non-zero values. In the second case a semi-sparse model was considered, so that half of the taps were different from zero. Finally, we explore the case where there is no sparsity in the system at all, so that all taps contribute equally to the computation of the output. The input signal follows a Gaussian distribution with a signal to noise ratio of 20 dB. A total of 1200 independent simulations were averaged to obtain the learning curves for each experiment. Each simulation had 4500 iterations, where the first 1500 corresponds to the first scenario described before, the next 1500 iterations corresponds to the second scenario and the last group of iterations comprised the third scenario. For the first experiment, we compare the performance of the conventional adaptive algorithms for the task of system identification. The results are shown in Figure 3.1.

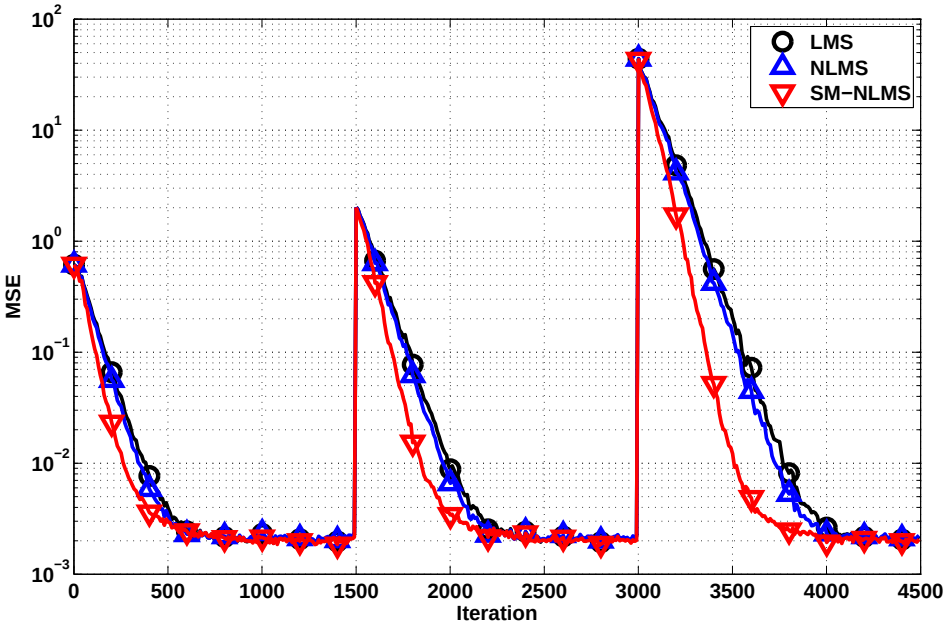

Figure 3.1: Adaptive Filtering Algorithms.

Next, we consider the performance of the LMS based algorithms. In this simulation we analyse three LMS algorithms each with a different penalty function. The results agree with the ones obtained in [34] and are shown in Figure 3.2.The fastest convergence and the lower MSE level was obtained by the EZA-LMS algorithm.

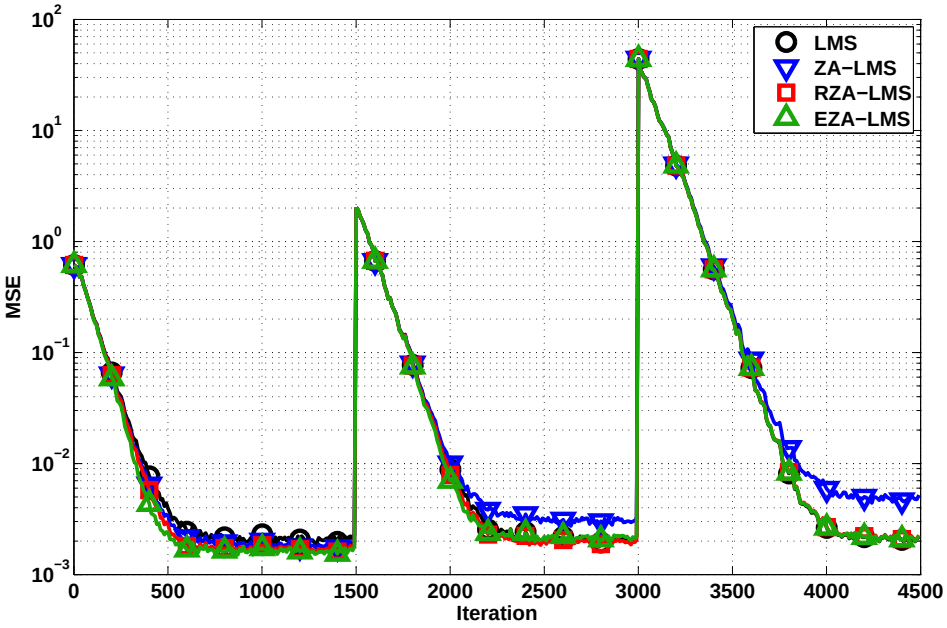

Figure 3.2: Performance of LMS Algorithms.

For the next experiment, we compare the performance of the NLMS based algorithms including the PNLMS algorithm, which was also developed to take into account the sparsity of a system. The results are shown in Figure 3.3. We see that the PNLMS algorithms has initially a very fast convergence speed but this speed decreases over time, and also the final MSE achieved by this algorithm is the worst as compared to the other algorithms developed for sparse systems.

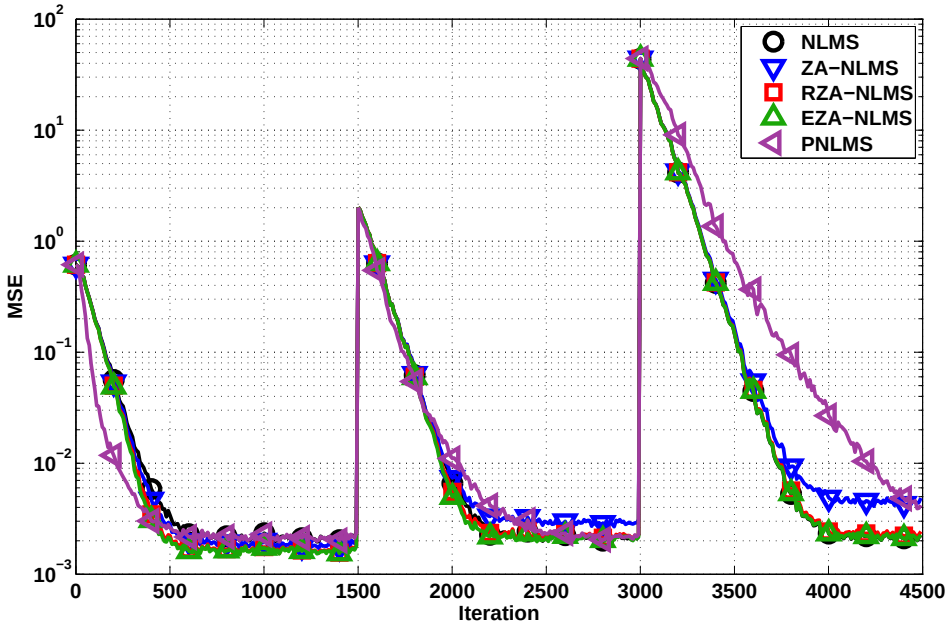

Figure 3.3: Performance of NLMS Algorithms.

The following experiment considers the performance of the SM-NLMS based algorithms. The results are shown in Figure 3.4.

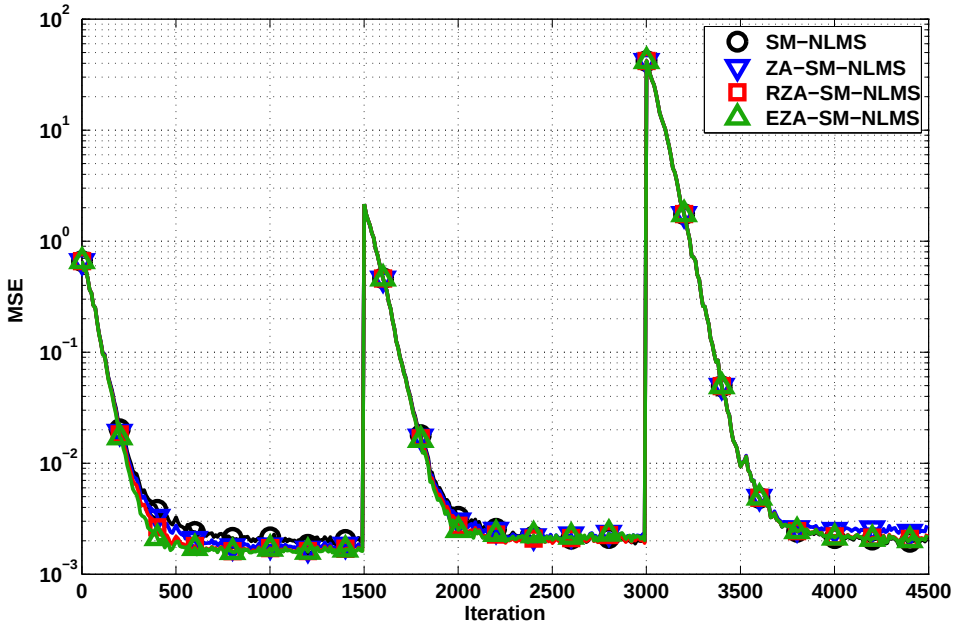

Figure 3.4: Performance of SM-NLMS Algorithms.

For the following experiment, we consider the performance of the proposed algorithms. Figure 3.5 shows the performance for the ZA-SM-NLMS algorithms, Figure 3.6 shows the results for the RZA-SM-NLMS algorithms and Figure 3.7 shows the performance of the EZA-SM-NLMS algorithms. The proposed algorithms achieve a faster convergence speed in sparse environments.

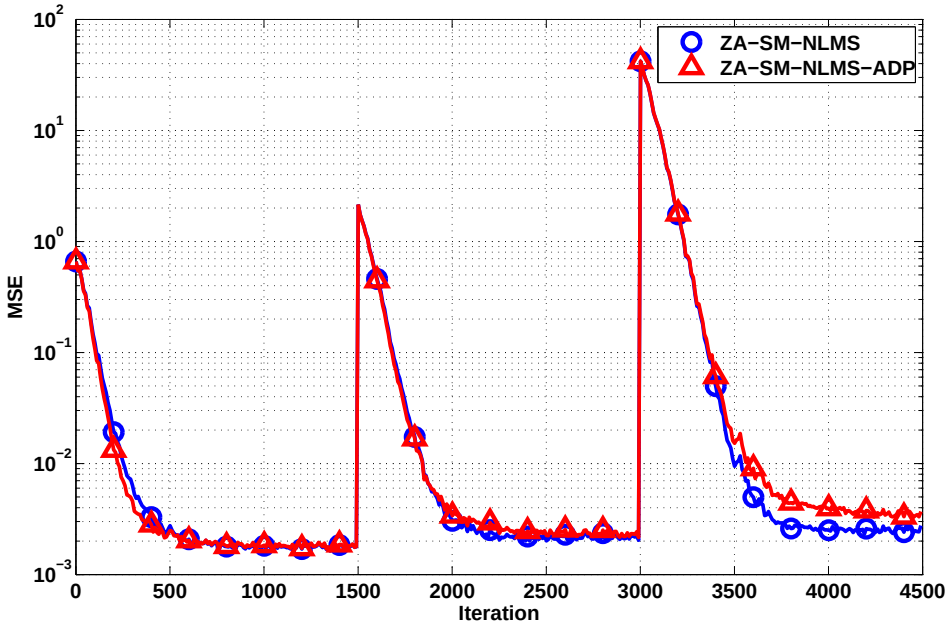

Figure 3.5: Performance of ZA-SM-NLMS Algorithms.

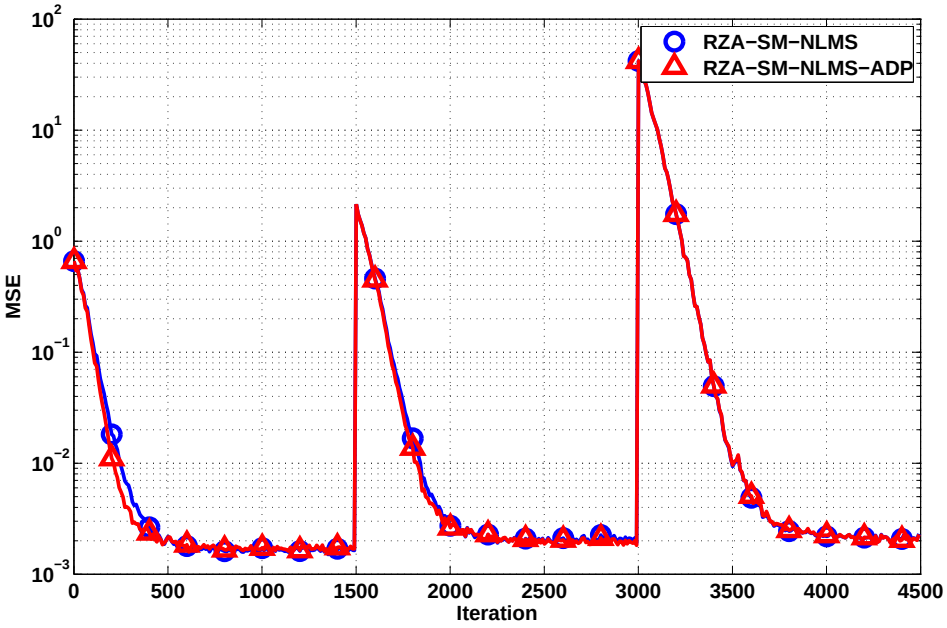

Figure 3.6: Performance of RZA-SM-NLMS Algorithms.

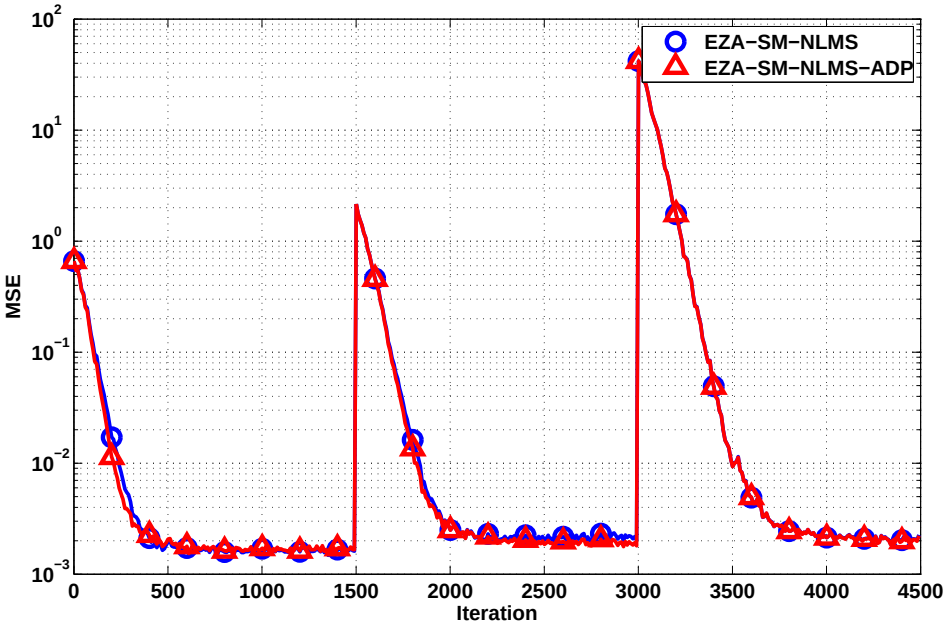

Figure 3.7: Performance of EZA-SM-NLMS Algorithms.

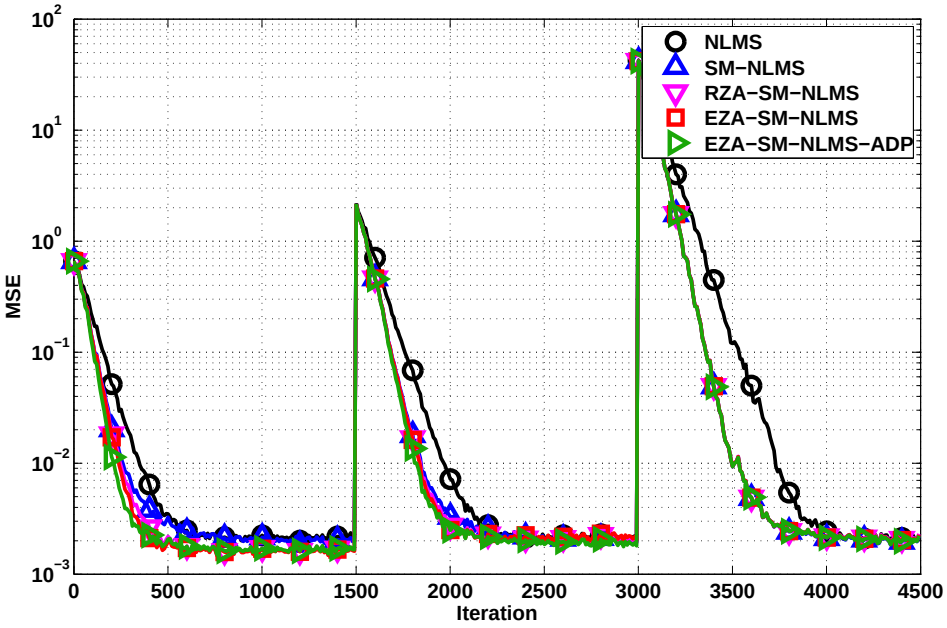

Figure 3.8: Performance of Adaptive Algorithms.

Table 3.2 summarizes the update rate achieved by the proposed algorithms in a sparse scenario, which is approximately 1% less than the results obtained by conventional adaptive algorithms.

Table 3.2: Update Rate

Algorithm	%
ZA-SM-NLMS-ADP	25.84%
RZA-SM-NLMS-ADP	23.67%
EZA-SM-NLMS-ADP	23.28%

Finally, we consider two different correlated inputs to evaluate the performance of the proposed EZA-SM-NLMS-ADP algorithm. The input is generated by a white Gaussian sequence $v(i)$, uncorrelated with the noise. Then this signal is passed through two different IIR filters described by

$$x_1(i) = 0.7x(i-1) + v(i) \tag{3-105}$$

$$x_4(i) = 0.8x(i-1) + 0.19x(i-2) + 0.09x(i-3) - 0.5x(i-4) + v(i), \tag{3-106}$$

which corresponds to first- and fourth-order autoregressive (AR) processes, respectively [55]. For the learning curves, we consider a total of 5000 iterations,

where the first 5000 iterations correspond to the sparse scenario and the last set of iterations represent the semi-sparse scenario. The results in Fig. 3.9 show that a correlated input slows the converge speed and increases the steady-state MSE. In such cases, applying a penalty function improves both results, the convergence speed and the steady-state MSE.

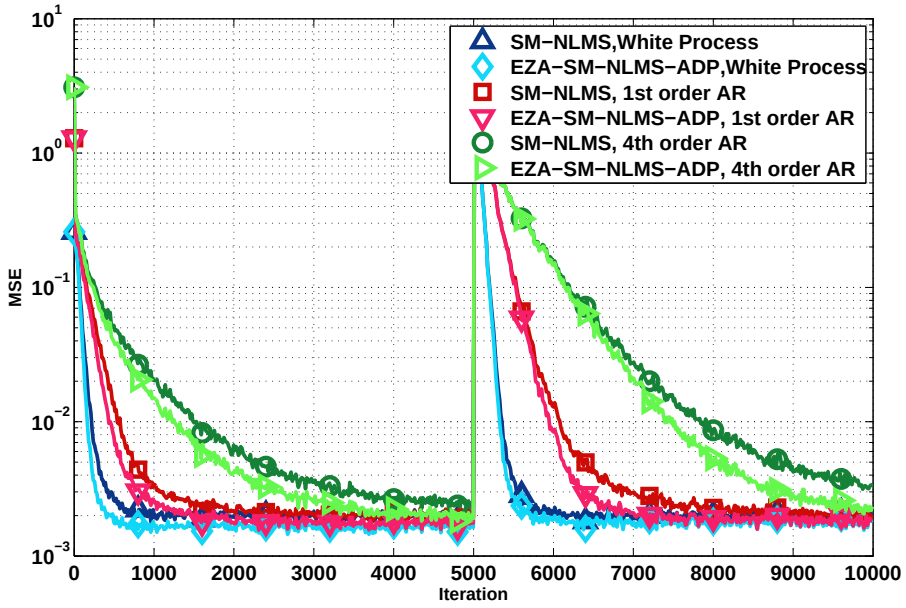

Figure 3.9: Performance of the EZA-SM-NLMS-ADP and the SM-NLMS algorithm with correlated inputs

3.3.2 Steady State MSE of the ZA-SM-NLMS algorithm

To test the equations obtained in Section 3.2.3 for the ZA-SM-NLMS algorithm, several experiments were performed. A system with 32 weights and with a degree of sparsity of 1/8 was used for the tests. The desired signal was corrupted by white Gaussian noise with $\sigma_n = 0.04$. The SNR was set to 20 dB for each experiment. The averaged MSE was computed over a total of 1000 simulations each one with 12000 iterations. For the first experiment we compute the probability of update for different error bounds. The results from Fig. 3.10 show that the probability of update converges to a constant at steady-state, which means that the assumption made in our development is reasonable.

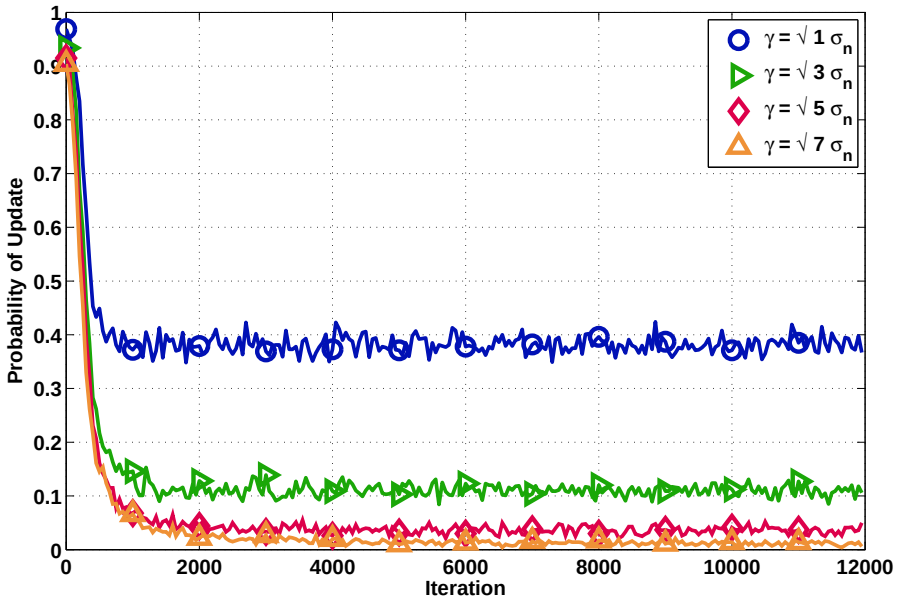

Figure 3.10: Probability of Update for different bounds.

Next, we measure the probability of update at steady state for different values of $\gamma = \sqrt{\tau}\sigma_n$ and compared it to equation (3-69). Figure 3.11 summarizes the results obtained.

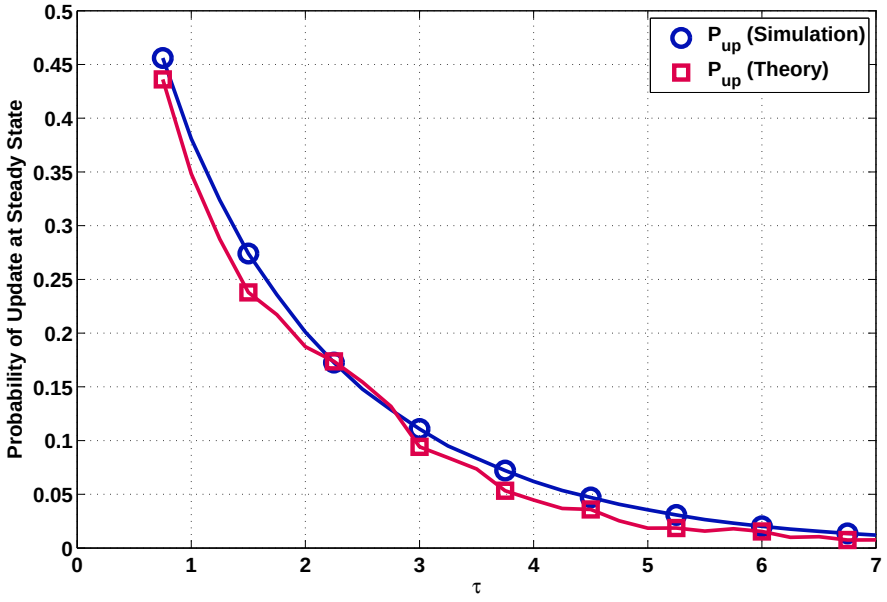

Figure 3.11: Probability of Update at Steady State.

Finally, we compare the theoretical MSE of the ZA-SM-NLMS algorithm with the MSE obtained by simulations. The results are shown in Figure 3.12 and Figure 3.13.

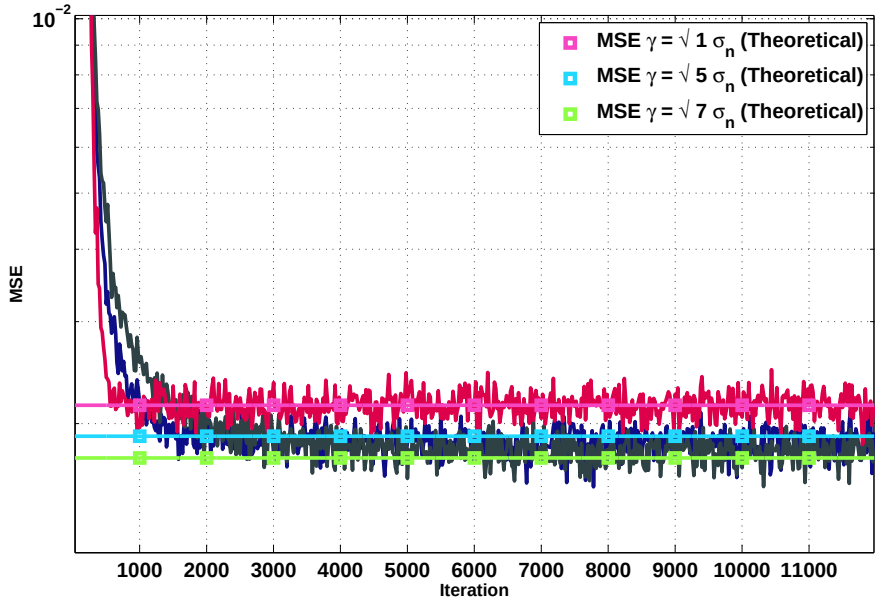

Figure 3.12: Steady-state MSE for the ZA-SM-NLMS algorithm.

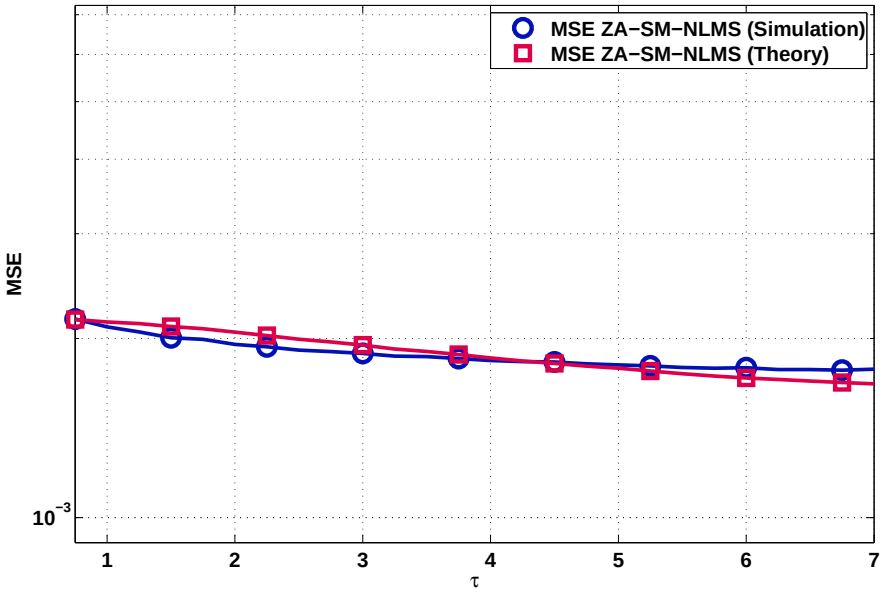

Figure 3.13: ZA-SM-NLMS MSE for different error bounds.

We can conclude that the mathematical model and expressions developed to describe the MSE of the ZA-SM-NLMS algorithm produce curves that match those obtained by simulation as evidenced by examples.

4
Data-Selective Kernel Adaptive Algorithms

In this chapter we develop data-selective kernel adaptive algorithms. The algorithms described herein use an adaptive step-size to achieve a better trade-off between convergence speed and steady state. The data-selective update scheme used also limits the size of the dictionary created by the kernel expansion, which is one of the major problems when using a kernel adaptive algorithm. A statistical analysis of the algorithm is also carried out. Finally, several experiments are performed to test the proposed algorithms.

4.1
Set-Membership Normalized Kernel Least-Mean-Squares

Let us consider an adaptive filtering problem with a sequence of training samples given by $\{\mathbf{x}(i), d(i)\}$, where $\mathbf{x}(i)$ is the N-dimensional input vector of the system and $d(i)$ represents the desired signal at time instant i. The output of the adaptive filter is given by

$$y(i) = \mathbf{w}^{\mathrm{T}}(i)\,\mathbf{x}(i), \tag{4-1}$$

where $\mathbf{w}(i)$ is the weight vector with length N.

Let us define a non-linear transformation denoted by $\varphi : \mathbb{R}^n \to \mathbb{F}$, that maps the input to a high-dimensional feature space. Applying the transformation stated before, we map the input and the weights to a high-dimensional space obtaining:

$$\boldsymbol{\varphi}(i) = \varphi(\mathbf{x}(i)), \tag{4-2}$$

$$\boldsymbol{\omega}(i) = \varphi(\mathbf{w}(i)). \tag{4-3}$$

We should emphasize that $\boldsymbol{\omega}(i)$ is now a vector where each component is a function of the elements of $\mathbf{w}(i)$, so that the dimension of $\boldsymbol{\omega}(i)$ is greater than $\mathbf{w}(i)$. The following equation describes the error generated by the system:

$$e(i) = d(i) - \boldsymbol{\omega}^{\mathrm{T}}(i)\,\boldsymbol{\varphi}(i). \tag{4-4}$$

The main objective of the proposed algorithm is to model a function, such that the magnitude of the estimated error defined by (4-4) is upper bounded by a quantity γ. Assuming that the value of γ is appropriately chosen, there

exist several functions that satisfy the error requirement. To summarize, any function leading to an estimation error smaller than the defined threshold is an adequate solution, resulting in a set of filters. Otherwise if the value of γ is not properly chosen (if it is too small for example), then there might be no solution.

Consider a set $\bar{\boldsymbol{S}}$ containing all the possible input-desired pairs $\{\boldsymbol{\varphi}(i), d(i)\}$ of interest. Now we can define a set $\boldsymbol{\theta}$ with all the possible functions leading to an estimation error bounded in magnitude by γ. This set is known as the feasibility set and is expressed by

$$\boldsymbol{\theta} = \bigcap_{\{\boldsymbol{\varphi},d\}\in\bar{\boldsymbol{S}}} \left\{\boldsymbol{\omega} \in \mathbb{F} \,/\, |d - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}| \leq \gamma\right\}. \tag{4-5}$$

Suppose that we are only interested in the case in which only measured data are available. Let us define a new set $\mathcal{H}(i)$ with all the functions such that the estimation error is upper bounded by γ . The set is called the constraint set and is mathematically defined by

$$\mathcal{H}(i) = \left\{\boldsymbol{\omega} \in \mathbb{F} \,/\, |d(i) - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(i)| \leq \gamma\right\}. \tag{4-6}$$

It follows from (4-6) that, for each data pair, there exists an associated constraint set. The set containing the intersection of the constraint sets over all available time instants is called exact membership set and is given by the following equation:

$$\psi(i) = \bigcap_{k=0}^{i} \mathcal{H}(i). \tag{4-7}$$

The exact membership set, $\psi(i)$, should become small as the data containing new information arrives. This means that at some point the adaptive filter will reach a state where $\psi(i) = \psi(i-1)$, so that there is no need to update $\boldsymbol{\omega}$. This happens because $\psi(i-1)$ is already a subset of $\mathcal{H}(i)$. As a result, the update of any set-membership based algorithm is data dependent, saving resources, a fact that is crucial in kernel based adaptive filters because of the growing structure that they create.

To develop the algorithm, we check first if the previous estimate is outside the constraint set, i.e., $|d(i) - \boldsymbol{\omega}^{\mathrm{T}}(i)\boldsymbol{\varphi}(i)| > \gamma$. If the error exceeds the bound established, the algorithm performs an update so that the *a posteriori* estimated error lies in $\mathcal{H}(i)$. In other words, an update should occur either if

$$e(i) = d(i) - \boldsymbol{\omega}^{\mathrm{T}}(i)\boldsymbol{\varphi}(i) > \gamma \tag{4-8}$$

or

$$e(i) = d(i) - \boldsymbol{\omega}^{\mathrm{T}}(i)\boldsymbol{\varphi}(i) < -\gamma \tag{4-9}$$

If any of the previous cases occurs we minimize $||\boldsymbol{\omega}(i+1) - \boldsymbol{\omega}(i)||^2$ subject to $\boldsymbol{\omega}(i+1) \in \mathcal{H}(i)$, which means that the *a posteriori* error $\xi_p(i)$ is given by

$$\xi_p(i) = d(i) - \boldsymbol{\omega}^{\mathrm{T}}(i+1)\,\boldsymbol{\varphi}(i) = \pm\gamma. \tag{4-10}$$

As mentioned in [13], the NKLMS update equation is given by

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \frac{\mu(i)}{\varepsilon + ||\boldsymbol{\varphi}(i)||^2} e(i)\,\boldsymbol{\varphi}(i), \tag{4-11}$$

where $\mu(i)$ is the step-size that should be chosen to satisfy the constraints of the algorithm and ε is a small constant used to avoid numerical problems. Substituting (4-11) in (4-10) we arrive at:

$$\xi_p(i) = d(i) - \boldsymbol{\omega}^{\mathrm{T}}(i)\,\boldsymbol{\varphi}(i) - \frac{\mu(i)}{\varepsilon + ||\boldsymbol{\varphi}(i)||^2} e(i)\,\boldsymbol{\varphi}^{\mathrm{T}}(i)\,\boldsymbol{\varphi}(i) = \pm\gamma. \tag{4-12}$$

Using (4-4) and replacing the dot products by kernel evaluations, the previous equation turns into:

$$\xi_p(i) = e(i) - \frac{\mu(i)}{\varepsilon + \kappa(\mathbf{x}(i), \mathbf{x}(i))} e(i)\,\kappa(\mathbf{x}(i), \mathbf{x}(i)) = \pm\gamma. \tag{4-13}$$

Assuming that the constant ε is sufficiently small to ensure that

$$\frac{\kappa(\mathbf{x}(i), \mathbf{x}(i))}{\varepsilon + \kappa(\mathbf{x}(i), \mathbf{x}(i))} \approx 1, \tag{4-14}$$

then, from equation (4-13), we have

$$\pm\gamma = e(i) - \mu(i)\,e(i) \tag{4-15}$$

$$= e(i)\,(1 - \mu(i)) \tag{4-16}$$

$$\gamma = |e(i)\,(1 - \mu(i))|. \tag{4-17}$$

If $\mu(i)$ takes values between 0 and 1, it follows that:

$$|e(i)|\,(1 - \mu(i)) = \gamma, \tag{4-18}$$

$$\mu(i) = 1 - \frac{\gamma}{|e(i)|}. \tag{4-19}$$

Taking into account that the update only occurs if the error is greater than the specified bound then $\mu(i)$ is fully described by

$$\mu(i) = \begin{cases} 1 - \frac{\gamma}{|e(i)|} & |e(i)| > \gamma, \\ 0 & \text{otherwise.} \end{cases} \tag{4-20}$$

We can compute $\boldsymbol{\omega}$ in (4-11) recursively as follows:

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i-1) + \frac{\mu(i-1)\, e(i-1)}{\varepsilon + ||\boldsymbol{\varphi}(i-1)||^2}\boldsymbol{\varphi}(i-1) + \frac{\mu(i)\, e(i)}{\varepsilon + ||\boldsymbol{\varphi}(i)||^2}\boldsymbol{\varphi}(i)$$

$$\vdots$$

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(0) + \sum_{k=1}^{i} \frac{\mu(k)}{\varepsilon + ||\boldsymbol{\varphi}(k)||^2} e(k)\, \boldsymbol{\varphi}(k) \tag{4-21}$$

Setting $\boldsymbol{\omega}(0)$ to zero leads to:

$$\boldsymbol{\omega}(i+1) = \sum_{k=1}^{i} \frac{\mu(k)}{\varepsilon + ||\boldsymbol{\varphi}(k)||^2} e(k)\, \boldsymbol{\varphi}(k). \tag{4-22}$$

The output of the filter to a new input $\boldsymbol{\varphi}(i+1)$ can be computed as the following inner product:

$$\begin{aligned}\boldsymbol{\omega}^{\mathrm{T}}(i+1)\,\boldsymbol{\varphi}(i+1) &= \left[\sum_{k=1}^{i} \frac{\mu(k)}{\varepsilon + ||\boldsymbol{\varphi}(k)||^2} e(k)\, \boldsymbol{\varphi}^{\mathrm{T}}(k)\right] \boldsymbol{\varphi}(i+1), \\ &= \sum_{k=1}^{i} \frac{\mu(k)}{\varepsilon + ||\boldsymbol{\varphi}(k)||^2} e(k)\, \boldsymbol{\varphi}^{\mathrm{T}}(k)\, \boldsymbol{\varphi}(i+1). \end{aligned} \tag{4-23}$$

Using the kernel trick we obtain:

$$\boldsymbol{\omega}^{\mathrm{T}}(i+1)\,\boldsymbol{\varphi}(i+1) = \sum_{k=1}^{i} \frac{\mu(k)\, e(k)}{\varepsilon + \kappa(\mathbf{x}(k), \mathbf{x}(k))} \kappa(\mathbf{x}(k), \mathbf{x}(i+1)), \tag{4-24}$$

where $\mu(k)$ is given by (4-20). Let us define a coefficient vector $\boldsymbol{a}(i)$ that stores the following product:

$$a_i(i) = \mu(i)\, e(i), \tag{4-25}$$

so that (4-24) becomes:

$$\boldsymbol{\omega}^{\mathrm{T}}(i+1)\,\boldsymbol{\varphi}(i+1) = \sum_{k=1}^{i} \frac{a_k(i)}{\varepsilon + \kappa(\mathbf{x}(k), \mathbf{x}(k))} \kappa(\mathbf{x}(k), \mathbf{x}(i+1)). \tag{4-26}$$

Equations (4-4), (4-20),(4-25), and (4-26) summarize the proposed Set-Membership Normalized Kernel Least Mean-Square (SM-NKLMS) algorithm. We set the initial value of $\boldsymbol{\omega}$ to zero as well as the first coefficient. As new inputs arrive we can calculate the output of the system with (4-26). Then the error may be computed with (4-4) and if it exceeds the bound established we compute the step-size with (4-20). Finally, we update the coefficients $\boldsymbol{a}(i)$ with (4-25). Note that some coefficients may be zero due the data selectivity characteristic of the algorithm. We do not need to store the zero coefficients as they do not contribute to the output calculus, resulting in a saving of resources. This means that the dictionary at time instant i, denoted by $\boldsymbol{C}(i)$, has only m

elements, with $m < i$. Each column of the dictionary, denoted by $\boldsymbol{c}_j$, contains the input that performed the kth update. We can now rewrite equation (4-26) as follows:

$$\boldsymbol{\omega}^{\mathrm{T}}(i+1)\,\boldsymbol{\varphi}(i+1) = \sum_{k=1}^{m} \frac{a_k(i)}{\varepsilon + \kappa(\boldsymbol{c}_k, \boldsymbol{c}_k)} \kappa(\mathbf{x}(i), \boldsymbol{c}_k) \tag{4-27}$$

This is an important result because it controls the growing network created by the algorithm. In stationary environments the algorithm will limit the growing structure. Algorithm 4 summarizes the SM-NKLMS algorithm.

Algorithm 4 Set-Membership Normalized Kernel Least Mean Squares

Initialization

1.Choose γ , ε and κ.

2.$\boldsymbol{C}(1) = \{\mathbf{x}(1)\}$

3.$\mu(1) = 1 - \frac{\gamma}{|d(1)|}$

4.$a_1(1) = \mu(1)\, d(1)$

5.$m = 1$

Computation

6.**while** $\{\mathbf{x}(i), d(i)\}$ available **do**:

%Compute the output

7. $f_{i-1}(\mathbf{x}(i)) = \sum_{k=1}^{m} \frac{a_k(i)}{\varepsilon+\kappa(\boldsymbol{c}_k,\boldsymbol{c}_k)} \kappa(\mathbf{x}(i), \boldsymbol{c}_k)$

%Compute the error

8. $e(i) = d(i) - f_{i-1}(\mathbf{x}(i))$

9. **if** $|e(i)| > \gamma$

%Compute the step-size

10. $\mu(i) = 1 - \frac{\gamma}{|e(i)|}$

%Update the coefficients

11. $$\boldsymbol{a}(i+1) = \begin{bmatrix} \boldsymbol{a}(i) \\ 0 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mu(i)\, e(i) \end{bmatrix}$$

%Store the new center

12. $\boldsymbol{\mathcal{C}}(i+1) = \{\boldsymbol{\mathcal{C}}(i), \mathbf{x}(i)\}$

13. $m = m + 1$

14. **else**

15. $\mu(i) = 0$

16. $\boldsymbol{a}(i+1) = \boldsymbol{a}(i)$

17. $\boldsymbol{\mathcal{C}}(i+1) = \boldsymbol{\mathcal{C}}(i)$

18. **end if**

19.**end while**

4.2

Nonlinear Regression approach

In this section, we follow a nonlinear regression approach as described in [48,66], to develop a data-selective algorithm based on a kernel function. Let us define a function $\psi(\cdot)$ on a feature space which, given an input vector $\mathbf{x}(i)$ generates the model output $\psi(\mathbf{x}(i))$. Our problem is now reduced to finding the function $\psi(\cdot)$ that minimizes the sum of the square error between the desired response and the model output as described by

$$\min_{\psi\in\mathcal{H}} \sum_{k=1}^{i} |d(k) - \psi(\mathbf{x}(k))|^2 \tag{4-28}$$

The representer theorem [53] states that the function $\psi(\cdot)$ can be expressed as a kernel expansion which depends on the available data, so that:

$$\psi(\cdot) = \sum_{k=1}^{i} a_k \kappa(\cdot, \mathbf{x}(k)). \tag{4-29}$$

Now we need to solve the following problem:

$$\min_{\boldsymbol{a}} \| \boldsymbol{d} - \boldsymbol{K}\boldsymbol{a} \|^2, \tag{4-30}$$

where $\boldsymbol{K}$ is the Gram matrix containing at each row i and each column j the kernel evaluations of the input data denoted by κ_{ij}, where:

$$[\boldsymbol{K}]_{ij} = \kappa_{ij} = \kappa(\mathbf{x}(i), \mathbf{x}(j)). \tag{4-31}$$

Let us now consider the case where we have a dictionary of size m. The vector $\mathbf{x}(\delta_j) = \boldsymbol{c}_j$ represents the jth element of the dictionary, where $\mathbf{x}(\delta_j) \neq \mathbf{x}(i)$ for $j = 1, \cdots, m$. Consider also a vector $\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)$, containing the kernel evaluations between the input data at time i and every input stored in the dictionary at time $i > m$, given by

$$\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i) = \begin{bmatrix} \kappa(\mathbf{x}(i), \mathbf{x}(\delta_1)) \\ \kappa(\mathbf{x}(i), \mathbf{x}(\delta_2)) \\ \vdots \\ \kappa(\mathbf{x}(i), \mathbf{x}(\delta_m)) \end{bmatrix}. \tag{4-32}$$

Using the instantaneous gradient to obtain the NKLMS algorithm, our problem becomes:

$$\begin{gathered} \min_{\boldsymbol{a}} \| \boldsymbol{a}(i+1) - \boldsymbol{a}(i) \|^2 \\ \text{subject to} \\ | d(i) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}{}^{\mathrm{T}}(i)\, \boldsymbol{a}(i+1) | = 0. \end{gathered} \tag{4-33}$$

Using the method of Lagrange multipliers, we have

$$\mathcal{L}(\boldsymbol{a}, \boldsymbol{\lambda}) = \| \boldsymbol{a}(i+1) - \boldsymbol{a}(i) \|^2 + \lambda \left(d(i) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}{}^{\mathrm{T}}(i)\, \boldsymbol{a}(i+1) \right). \tag{4-34}$$

Calculating the gradient with respect to to $\boldsymbol{a}(i+1))$ and λ, we obtain

$$\frac{\partial \mathcal{L}(\boldsymbol{a}, \boldsymbol{\lambda})}{\partial \boldsymbol{a}(i+1)} = (\boldsymbol{a}(i+1) - \boldsymbol{a}(i)) - \lambda \boldsymbol{\kappa_\delta}(i) = \mathbf{0}, \tag{4-35}$$

$$\frac{\partial \mathcal{L}(\boldsymbol{a}, \boldsymbol{\lambda})}{\partial \lambda} = d(i) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i)\, \boldsymbol{a}(i+1) = 0. \tag{4-36}$$

From equation (4-35) we obtain:

$$\lambda \boldsymbol{\kappa_\delta}(i) = (\boldsymbol{a}(i+1) - \boldsymbol{a}(i)), \tag{4-37}$$

$$\lambda \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i)\, \boldsymbol{\kappa_\delta}(i) = \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i)\, (\boldsymbol{a}(i+1) - \boldsymbol{a}(i)). \tag{4-38}$$

Substituting (4-36) in the equation above we get:

$$\lambda \parallel \boldsymbol{\kappa_\delta}(i) \parallel^2 = \left(d(i) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i)\, \boldsymbol{a}(i)\right), \tag{4-39}$$

$$\lambda = \frac{1}{\parallel \boldsymbol{\kappa_\delta}(i) \parallel^2} \left(d(i) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i)\, \boldsymbol{a}(i)\right). \tag{4-40}$$

Finally, replacing λ in equation (4-35) we obtain the NKLMS update recursion for the coefficients, which is expressed as follows:

$$\boldsymbol{a}(i+1) = \boldsymbol{a}(i) + \frac{1}{\parallel \boldsymbol{\kappa_\delta}(i) \parallel^2} \left(d(i) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i)\, \boldsymbol{a}(i)\right) \boldsymbol{\kappa_\delta}(i). \tag{4-41}$$

When using the SM-NKLMS algorithm, the update only occurs when the error represented by $d(n) - \boldsymbol{\kappa_\delta}(i)^{\mathrm{T}}(n)\, \boldsymbol{a}(n)$ exceeds the threshold γ. In this case, the dictionary size should be increased by one as well as the length of the vector $\boldsymbol{a}$. The update recursion is given by

$$\boldsymbol{a}(i+1) = \begin{bmatrix} \boldsymbol{a}(i) \\ 0 \end{bmatrix} + \frac{\mu(i)}{\varepsilon + \parallel \boldsymbol{\kappa_\delta}(i) \parallel^2} \left(d(i) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i) \begin{bmatrix} \boldsymbol{a}(i) \\ 0 \end{bmatrix}\right) \boldsymbol{\kappa_\delta}(i). \tag{4-42}$$

Let us now define the *a posteriori* error as follows:

$$\xi_p(i) = d(i) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i)\, \boldsymbol{a}(i+1) = \pm\gamma. \tag{4-43}$$

Substituting equation (4-42) in the last equation and assuming that $\frac{\kappa(\mathbf{x}(i),\mathbf{x}(i))}{\varepsilon+\kappa(\mathbf{x}(i),\mathbf{x}(i))} \approx 1$, we have

$$d(i) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i) \begin{bmatrix} \boldsymbol{a}(i) \\ 0 \end{bmatrix} - \mu(i) \left(d(i) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i) \begin{bmatrix} \boldsymbol{a}(i) \\ 0 \end{bmatrix}\right) = \pm\gamma. \tag{4-44}$$

Simplifying terms, we obtain

$$\begin{aligned} \gamma =& e(i) - \mu(i)\, e(i), \\ =& e(i)\, (1 - \mu(i)). \end{aligned} \tag{4-45}$$

From the last equation we obtain an expression for the step-size, wich is

given by

$$\mu(i) = \begin{cases} 1 - \frac{\gamma}{|e(i)|} & |\, e(i) \,| > \gamma, \\ 0 & \text{otherwise.} \end{cases} \tag{4-46}$$

If the error does not exceed the threshold γ, the size of the dictionary remains the same and no coefficients update is performed, only the output of the model is calculated for the new input. The pseudo-code for this method is shown in Algorithm 5.

Algorithm 5 Nonlinear Regression SM-NKLMS Algorithm

Initialization

1.Choose γ , ε and κ.

2.$\mu(1) = 1 - \frac{\gamma}{|d(1)|}$

3.$\boldsymbol{a}(1) = 0$

4.$m = 1$

5.$\boldsymbol{\kappa_\delta}(1) = \kappa(\mathbf{x}(1), \mathbf{x}(1))$

Computation

6.**while** $\{\mathbf{x}(i), d(i)\}$ available **do**:

%Compute vector $\boldsymbol{\kappa_\delta}(i)$

7. $\boldsymbol{\kappa_\delta}(i) = \{\kappa(\mathbf{x}(i), \mathbf{x}(\delta_1)), \ldots, \kappa(\mathbf{x}(i), \mathbf{x}(\delta_m))\}$

%Compute the output

8. $y(i) = \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i)\, \boldsymbol{a}(i)$

%Compute the error

9. $e(i) = d(i) - y(i)$

10. **if** $|e(i)| > \gamma$

%Store the new center

11. $\boldsymbol{\kappa_\delta}(i) = \{\kappa(\mathbf{x}(i), \mathbf{x}(\delta_1)), \ldots, \kappa(\mathbf{x}(i), \mathbf{x}(\delta_{m+1}))\}$

%Store the step-size

12. $\mu(i) = 1 - \frac{\gamma}{|e(i)|}$

%Update the coefficients

13. $\boldsymbol{a}(i+1) = \begin{bmatrix} \boldsymbol{a}(i) \\ 0 \end{bmatrix} + \frac{\mu(i)}{\varepsilon + \|\boldsymbol{\kappa_\delta}(i)\|^2} \left(d(i) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i) \begin{bmatrix} \boldsymbol{a}(i) \\ 0 \end{bmatrix} \right) \boldsymbol{\kappa_\delta}(i)$

14. $m = m + 1$

15. **else**

16. $\mu(i) = 0$

17. $\boldsymbol{a}(i+1) = \boldsymbol{a}(i)$

18. **end if**

19.**end while**

4.3

Set Membership-Kernel Affine Projection Algorithm

As described in Subsection 2.4.2, the KAP algorithm uses the last K inputs to update the coefficients. Based on this fact, let us consider the past K constraint sets to perform the update. Under this scope, it is also convenient to redefine the exact membership as follows:

$$\psi(i) = \left(\bigcap_{j=0}^{i-K} \mathcal{H}(j)\right)\left(\bigcap_{l=i-K+1}^{i} \mathcal{H}(l)\right) = \psi^{i-K}(i) \bigcap \psi^{K}(i), \tag{4-47}$$

where $\psi^{K}(i)$ emphasizes the use of K constraint sets for updating. This means that vector $\boldsymbol{\omega}(i)$ should belong to $\psi^{K}(i)$. In order to develop the Set-Membership Kernel Affine Projection (SM-KAP) algorithm, we need to set several bounds $\bar{\gamma}_k(i)$, for $k = 1, \ldots, K$, so that the error magnitudes should satisfy this constraints after updating. It follows that there exists a space $S(i-k+1)$ containing all vectors $\boldsymbol{\omega}$ satisfying the following expression:

$$d(i-k+1) - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(i-k+1) = \bar{\gamma}_k(i), \quad \text{for } k = 1, \ldots, K. \tag{4-48}$$

The SM-KAP algorithm should perform an update whenever $\boldsymbol{\omega}(i) \notin \psi^{K}(i)$, so that the following constrained optimization problem is solved:

$$\begin{aligned} &\min_{\boldsymbol{\omega}(i+1)} \| \boldsymbol{\omega}(i+1) - \boldsymbol{\omega}(i) \|^2 \\ &\text{subject to} \\ &\boldsymbol{d}(i) - \boldsymbol{\Phi}^{\mathrm{T}}(i)\boldsymbol{\omega}(i+1) = \bar{\boldsymbol{\gamma}}(i), \end{aligned} \tag{4-49}$$

where $\bar{\boldsymbol{\gamma}}(i)$ is a vector containing all the K bounds. Modifying the optimization problem in (4-49) with the method of the Lagrange multipliers we get

$$\mathcal{L}(\boldsymbol{\omega}(i)) = \| \boldsymbol{\omega}(i+1) - \boldsymbol{\omega}(i) \|^2 + \boldsymbol{\lambda}^{\mathrm{T}}(i)\left(\boldsymbol{d}(i) - \boldsymbol{\Phi}^{\mathrm{T}}(i)\boldsymbol{\omega}(i+1) - \bar{\boldsymbol{\gamma}}(i)\right), \tag{4-50}$$

where $\boldsymbol{\lambda}^{\mathrm{T}}(i)$ is the vector of Lagrange multipliers. Now we can compute the gradient of $\mathcal{L}(\boldsymbol{\omega}(i))$ and equate it to zero, which yields

$$\frac{\partial \mathcal{L}(\boldsymbol{\omega}(i))}{\partial \boldsymbol{\omega}(i)} = 2\boldsymbol{\omega}(i+1) - 2\boldsymbol{\omega}(i) - \boldsymbol{\lambda}^{\mathrm{T}}(i)\boldsymbol{\Phi}^{\mathrm{T}}(i) = \mathbf{0}. \tag{4-51}$$

Solving for $\boldsymbol{\omega}(i+1)$, we obtain

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \sum_{k=1}^{K} \frac{\lambda_k(i)}{2}\boldsymbol{\varphi}(i-k+1), \tag{4-52}$$

$$= \boldsymbol{\omega}(i) + \frac{1}{2}\boldsymbol{\Phi}(i)\boldsymbol{\lambda}(i). \tag{4-53}$$

From equation (4-49), we know that

$$\boldsymbol{d}(i) - \bar{\boldsymbol{\gamma}}(i) = \boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\omega}(i+1). \tag{4-54}$$

Substituting equation (4-53) in (4-54) we have

$$\begin{aligned}
\boldsymbol{d}(i) - \bar{\boldsymbol{\gamma}}(i) &= \boldsymbol{\Phi}^{\mathrm{T}}(i)\left(\boldsymbol{\omega}(i) + \frac{1}{2}\boldsymbol{\Phi}(i)\,\boldsymbol{\lambda}(i)\right), \\
&= \boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\omega}(i) + \boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\Phi}(i)\,\frac{\boldsymbol{\lambda}(i)}{2}, \qquad (4\text{-}55) \\
\boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\Phi}(i)\,\frac{\boldsymbol{\lambda}(i)}{2} &= \boldsymbol{d}(i) - \boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\omega}(i) - \bar{\boldsymbol{\gamma}}(i), \\
&= \boldsymbol{e}(i) - \bar{\boldsymbol{\gamma}}(i), \qquad (4\text{-}56) \\
\frac{\boldsymbol{\lambda}(i)}{2} &= \left(\boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\Phi}(i)\right)^{-1}\left(\boldsymbol{e}(i) - \bar{\boldsymbol{\gamma}}(i)\right). \qquad (4\text{-}57)
\end{aligned}$$

We can now obtain the update equation, which is used as long as the error is greater than the established bound, i.e., $|e(i)| > \bar{\gamma}$

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \boldsymbol{\Phi}(i)\left(\boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\Phi}(i)\right)^{-1}\left(\boldsymbol{e}(i) - \bar{\boldsymbol{\gamma}}(i)\right) \tag{4-58}$$

We have to consider that the error vector $\boldsymbol{e}(i)$ is composed by the actual error and $K-1$ *a posterior* errors, corresponding to the $K-1$ last inputs used

$$\boldsymbol{e}(i) = \begin{bmatrix} e(i) & e_p(i-1) & \cdots & e_p(i-K+1) \end{bmatrix}, \tag{4-59}$$

where $e_p(i-k)$ denotes the *a posteriori* error computed using the coefficients at iteration i as described by

$$e_p(i-k) = d(i-k) - \boldsymbol{\varphi}^{\mathrm{T}}(i-k)\,\boldsymbol{\omega}(i). \tag{4-60}$$

We will now discuss how to choose the vector $\bar{\boldsymbol{\gamma}}(i)$. A simple choice would be to force all the last K *a posteriori* errors to zero, which we term Zero Forcing (ZF) technique, i.e., $\bar{\boldsymbol{\gamma}}(i) = \mathbf{0}$. By replacing this condition on equation (4-58), we obtain

$$\boldsymbol{\omega}(i+1) = \begin{cases} \boldsymbol{\omega}(i) + \boldsymbol{\Phi}(i)\left(\boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\Phi}(i)\right)^{-1}\boldsymbol{e}(i) & |e(i)| > \bar{\gamma}, \\ \boldsymbol{\omega}(i) & \text{other case.} \end{cases} \tag{4-61}$$

We can verify that the equation above is the same as equation (2-65) used for the KAP algorithm but it uses a unit step-size. The main difference of this algorithm is that the update only takes place if $\boldsymbol{\omega}(i) \notin \mathcal{H}(k)$, which means that $|e(i)| > \bar{\gamma}$. This leads to a reduction in the computational complexity due the data selectivity. Using equation (2-70) to redefine the update equation we have

$$\boldsymbol{\omega}(i+1) = \textstyle\sum_{j=1}^{i-1} a_j(i)\,\boldsymbol{\varphi}(j) + \boldsymbol{\Phi}(i)\,\tilde{\mathbf{a}}(i) \quad |e(i)| > \bar{\gamma}\ . \tag{4-62}$$

From the equation above, it follows that the coefficients are only updated if $|e(i)| > \bar{\gamma}$, leading to

$$a_k(i) = \begin{cases} \tilde{a}_k(i), & k = i, \\ a_k(i-1) + \tilde{a}_{K+k-i}(i), & i-K+1 \leq k \leq i-1, \\ a_k(i-1) & 1 \leq k < i-K+1, \end{cases} \tag{4-63}$$

$$a_k(i) = a_k(i-1) \quad |e(i)| < \bar{\gamma} \ . \tag{4-64}$$

The SM-KAP-ZF algorithm is summarized in algorithm 6

Algorithm 6 SM-KAP-ZF

Initialization

1.Choose $\bar{\gamma}$, ε, K and κ.

2.$\boldsymbol{C}(1) = \{\mathbf{x}(1)\}, a_1(1) = d(1), m = 1$

Computation

3.**while** $\{\mathbf{x}(i), d(i)\}$ available **do**:

4. **for** $n = 1$ to $\min(i, K)$ **do**

%Compute the outputs

5. $y_n(i) = \sum_{k=1}^{m} a_k(i)\, \kappa(\mathbf{x}(i-n+1), \boldsymbol{c}_k)$

6. **end for**

%Compute the last K errors

7. $\boldsymbol{e}(i) = \boldsymbol{d}(i) - \boldsymbol{y}(i)$

8. **if** $|e(i)| > \bar{\gamma}$ **then**

9. **for** $n = 1$ to $\min(i, K)$ **do**

10. **for** $m = 1$ to $\min(i, K)$ **do**

%Compute the $K \times K$ matrix $\tilde{\mathbf{A}}$

11. $\tilde{A}_{n,m} = \kappa(\mathbf{x}(i-n+1), \mathbf{x}(i-m+1))$

12. **end for**

13. **end for**

14. $\tilde{\mathbf{A}} = (\tilde{\mathbf{A}} + \epsilon \mathbf{I})^{-1}$

%Update the K most recent coefficients

15. $\{\boldsymbol{a}(i)\}_{k=i-K+1}^{K} = \{\boldsymbol{a}(i-1)\}_{k=i-K+1}^{K} + \tilde{\mathbf{A}}\boldsymbol{e}(i)$

%Store the new center

16. $\boldsymbol{\mathcal{C}}(m+1) = \{\boldsymbol{\mathcal{C}}(m), \mathbf{x}(i)\}$

17. $m = m + 1$

18. **else**

19. $\boldsymbol{a}(i) = \boldsymbol{a}(i-1)$

20. **end if**

21.**end while**

Let us now consider another simple choice for vector $\bar{\boldsymbol{\gamma}}(i)$. We can exploit

the fact that the *a posteriori* error was updated to satisfy the constraint in equation (4-49). That means that we can set the values of $\bar{\gamma}_k(i)$ equal to $e_p(i-k+1)$ for $i \neq 1$. Substituting this condition in (4-58) we obtain

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \boldsymbol{\Phi}(i)\left(\boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\Phi}(i)\right)^{-1}(e(i) - \bar{\gamma}_1(i))\,\boldsymbol{u}, \tag{4-65}$$

where $\boldsymbol{u} = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^{\mathrm{T}}$. We can now select $\bar{\gamma_1}(i)$ as in the SM-NKLMS so that

$$\bar{\gamma_1}(i) = \bar{\gamma}\frac{e(i)}{|e(i)|}. \tag{4-66}$$

Replacing equation (4-66) in (4-65), we get

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \boldsymbol{\Phi}(i)\left(\boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\Phi}(i)\right)^{-1}(\mu(i)\,e(i))\,\boldsymbol{u}, \tag{4-67}$$

where

$$\mu(i) = \begin{cases} 1 - \frac{\bar{\gamma}}{|e(i)|} & |e(i)| > \bar{\gamma}, \\ 0 & \text{otherwise.} \end{cases} \tag{4-68}$$

Using equation (2-70) in (4-67) results in

$$\boldsymbol{\omega}(i+1) = \sum_{j=1}^{i-1} a_j(i-1)\,\boldsymbol{\varphi}(j) + (\mu(i)\,e(i))\,\boldsymbol{\Phi}(i)\,\tilde{\mathbf{a}}(i), \tag{4-69}$$

where the vector $\tilde{\mathbf{a}}(i)$ was redefined as

$$\tilde{\mathbf{a}}(i) = \left(\boldsymbol{\Phi}^{\mathrm{T}}(i)\,\boldsymbol{\Phi}(i) + \epsilon\mathbf{I}\right)^{-1}\boldsymbol{u}. \tag{4-70}$$

Finally, the coefficients are given by

$$a_k(i) = \begin{cases} \mu(i)\,e(i)\,\tilde{a}_k(i), & k = i, \\ a_k(i-1) + \mu(i)\,e(i)\,\tilde{a}_{K+k-i}(i), & i-K+1 \leq k \leq i-1, \\ a_k(i-1) & 1 \leq k < i-K+1. \end{cases} \tag{4-71}$$

The proposed SM-KAP is summarized in Algorithm 7.

Algorithm 7 SM-KAP

Initialization

1.Choose $\bar{\gamma}$, ε,K and κ.

2.$\boldsymbol{C}(1) = \{\mathbf{x}(1)\}$

3.$a_1(1) = \mu(1)\, d(1)$

Computation

4.**while** $\{\mathbf{x}(i), d(i)\}$ available **do**:

5. **for** $n = 1$ to $\min(i, K)$ **do**

%Compute the outputs

6. $y_n(i) = \sum_{k=1}^{m} a_k(i)\, \kappa(\mathbf{x}(i-n+1), \boldsymbol{c}_k)$

7. **end for**

%Compute the last K errors

8. $\boldsymbol{e}(i) = \boldsymbol{d}(i) - \boldsymbol{y}(i)$

9. **if** $|e(i)| > \bar{\gamma}$ **then**

%Compute the step-size

10. $\mu(i) = 1 - \frac{\bar{\gamma}}{|e(i)|}$

11. **for** $n = 1$ to $\min(i, K)$ **do**

12. **for** $m = 1$ to $\min(i, K)$ **do**

%Compute the $K \times K$ matrix $\tilde{\mathbf{A}}$

13. $\tilde{A}_{n,m} = \kappa(\mathbf{x}(i-n+1), \mathbf{x}(i-m+1))$

14. **end for**

15. **end for**

16. $\tilde{\mathbf{A}} = (\tilde{\mathbf{A}} + \epsilon \mathbf{I})^{-1}$

%Update the K most recent coefficients

17. $\{\boldsymbol{a}(i)\}_{k=i-K+1}^{K} = \{\boldsymbol{a}(i-1)\}_{k=i-K+1}^{K} + (\mu(i)\, e(i))\, \tilde{\mathbf{A}}\boldsymbol{u}$

%Store the new center

18. $\boldsymbol{\mathcal{C}}(m+1) = \{\boldsymbol{\mathcal{C}}(m), \mathbf{x}(i)\}$

19. $m = m + 1$

20. **else**

21. $\boldsymbol{a}(i) = \boldsymbol{a}(i-1)$

22. **end if**

23.**end while**

4.4 Statistical Analysis

Let us consider the nonlinear regression SM-NKLMS algorithm with a Gaussian kernel in a stationary environment, which means that $\varphi(\mathbf{x}(i))$ is stationary for $\mathbf{x}(i)$ stationary [67]. Several nonlinear systems used to model practical situations, such as Wiener and Hammerstein systems, satisfy this

assumption. The system inputs are N-dimensional, independent and identically distributed Gaussian vectors $\mathbf{x}(i)$ with zero-mean and variance equal to σ_x^2. Let us denote the autocorrelation matrix of the input vectors by $\mathbf{R}_{xx} = \mathbb{E}\left[\mathbf{x}(i)\,\mathbf{x}^{\mathrm{T}}(i)\right]$, so that $\mathbb{E}\left[\mathbf{x}(i-k)\,\mathbf{x}^{\mathrm{T}}(i-l)\right] = \mathbf{0}$ for $k \neq l$. However the components of the input vector can be correlated. Let us also consider a dictionary of fixed size M and the vector $\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)$ previously defined in equation (4-32). We consider that the vectors constituting the dictionary may change at each iteration following some dictionary updating scheme. The only limitation imposed is that $\mathbf{x}(\delta_j) \neq \mathbf{x}(\delta_k)$ for $j \neq k$, so that the dictionary vectors are statistically independent. The estimated output of the system is

$$y(i) = \boldsymbol{a}^{\mathrm{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i). \tag{4-72}$$

The corresponding estimation error is given by

$$e(i) = d(i) - y(i). \tag{4-73}$$

Squaring the equation above and taking the expected value results in the MSE:

$$\begin{aligned} J_{ms}(i) &= \mathbb{E}\left[e^2(i)\right] && (4\text{-}74)\\ &= \mathbb{E}\left[d^2(i)\right] - 2\mathbf{p}_{kd}^{\mathrm{T}}\boldsymbol{a}(i) + \boldsymbol{a}^{\mathrm{T}}(i)\,\mathbf{R}_{kk}\boldsymbol{a}(i), && (4\text{-}75)\end{aligned}$$

where $\mathbf{R}_{kk} = \mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}{}^{\mathrm{T}}(i)\right]$ represents the correlation matrix of the kernelized input, and $\mathbf{p}_{kd} = \mathbb{E}\left[d(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\right]$ is the cross-correlation vector between $\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)$ and $d(i)$. In [68, 69] it is shown that $\mathbf{R}_{kk}$ is positive definite. Thus, the Wiener solution and the minimum MSE are obtained as follows:

$$\begin{aligned} \boldsymbol{a}_o &= \mathbf{R}_{kk}^{-1}\mathbf{p}_{kd} && (4\text{-}76)\\ J_{min} &= \mathbb{E}\left[d^2(i)\right] - \mathbf{p}_{kd}^{\mathrm{T}}\mathbf{R}_{kk}^{-1}\mathbf{p}_{kd}, && (4\text{-}77)\end{aligned}$$

The entries of the correlation matrix $\mathbf{R}_{kk}$ are given by

$$\left[\mathbf{R}_{kk}\right]_{jl} = \begin{cases} \mathbb{E}\left[\kappa^2\left(\mathbf{x}(i), \mathbf{x}(\delta_j)\right)\right] & j = l \\ \mathbb{E}\left[\kappa\left(\mathbf{x}(i), \mathbf{x}(\delta_j)\right)\kappa\left(\mathbf{x}(i), \mathbf{x}(\delta_l)\right)\right] & j \neq l \end{cases} \tag{4-78}$$

Let us define the following products:

$$\left\|\mathbf{x}(i) - \mathbf{x}(\delta_j)\right\|^2 = \boldsymbol{y}_2^{\mathrm{T}}\mathbf{Q}_2\boldsymbol{y}_2 \tag{4-79}$$

$$\left\|\mathbf{x}(i) - \mathbf{x}(\delta_j)\right\|^2 + \left\|\mathbf{x}(i) - \mathbf{x}(\delta_l)\right\|^2 = \boldsymbol{y}_3^{\mathrm{T}}\mathbf{Q}_3\boldsymbol{y}_3, \tag{4-80}$$

where

$$\boldsymbol{y}_2 = \begin{bmatrix} \mathbf{x}^{\mathrm{T}}(i) & \mathbf{x}^{\mathrm{T}}(\delta_j) \end{bmatrix}^{\mathrm{T}}, \tag{4-81}$$

$$\boldsymbol{y}_3 = \left[\begin{array}{ccc} \mathbf{x}^{\mathrm{T}}(i) & \mathbf{x}^{\mathrm{T}}(\delta_j) & \mathbf{x}^{\mathrm{T}}(\delta_l) \end{array} \right]^{\mathrm{T}}, \tag{4-82}$$

$$\mathbf{Q}_2 = \begin{bmatrix} \mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{I} \end{bmatrix}, \tag{4-83}$$

$$\mathbf{Q}_3 = \begin{bmatrix} 2\mathbf{I} & -\mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{4-84}$$

We know from [68,70] that the moment generating a function of quadratic form $z = \boldsymbol{y}^{\mathrm{T}}\mathbf{Q}\boldsymbol{y}$, where $\boldsymbol{y}$ is a zero-mean Gaussian vector with covariance matrix $\mathbf{R}_y$ is given by

$$\mathbb{E}\left[e^{sz}\right] = \det\left\{\mathbf{I} - 2s\mathbf{Q}\mathbf{R}_y\right\}^{-\frac{1}{2}}. \tag{4-85}$$

The last equation allows us to compute the entries of the correlation matrix $\mathbf{R}_{kk}$ for the Gaussian kernel. Each element is given by

$$\left[\mathbf{R}_{kk}\right]_{jl} = \begin{cases} r_{md} = \det\left\{\mathbf{I}_2 - 2\mathbf{Q}_2\mathbf{R}_2/\nu^2\right\}^{-\frac{1}{2}} & j = l \\ r_{od} = \det\left\{\mathbf{I}_3 - \mathbf{Q}_3\mathbf{R}_3/\nu^2\right\}^{-\frac{1}{2}} & j \neq l \end{cases}. \tag{4-86}$$

4.4.1 Gaussian kernel SM-NKLMS algorithm transient behavior analysis

Let us define the coefficients-error vector defined by

$$\boldsymbol{v}(i) = \boldsymbol{a}(i) - \boldsymbol{a}_o. \tag{4-87}$$

The second-order moments of the coefficients are related to the MSE through [54]

$$J_{ms}(i) = J_{min} + tr\left\{\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}(i)\right\}, \tag{4-88}$$

where $\mathbf{C}_{\boldsymbol{v}}(i) = \mathbb{E}\left[\boldsymbol{v}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\right]$. This means that for studying the MSE behavior we need a model for $\mathbf{C}_{\boldsymbol{v}}(i)$. In this section, we derive an analytical model that describes the behavior of $\mathbf{C}_{\boldsymbol{v}}(i)$ for the proposed SM-NKLMS algorithm.

The coefficients update equation for the system is

$$\boldsymbol{a}(i+1) = \boldsymbol{a}(i) + \mu(i)\,e(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i), \tag{4-89}$$

where

$$\mu(i) = \begin{cases} 1 - \frac{\gamma}{|e(i)|} & |e(i)| > \gamma, \\ 0 & otherwise. \end{cases} \tag{4-90}$$

Subtracting $\boldsymbol{a_o}$ from equation (4-89), we obtain the weight error vector update equation:

$$\boldsymbol{v}(i+1) = \boldsymbol{v}(i) + \mu(i)\,e(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i). \tag{4-91}$$

The estimation error may now be rewritten as follows:

$$\begin{aligned} e(i) =& d(i) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{a}(i) \\ =& d(i) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{v}(i) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{a}_o. \end{aligned} \tag{4-92}$$

The optimum error is given by

$$e_o(i) = d(i) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{a}_o. \tag{4-93}$$

It follows that

$$e(i) = e_o(i) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{v}(i). \tag{4-94}$$

We may represent equation (4-89) by

$$\boldsymbol{a}(i+1) = \boldsymbol{a}(i) + P_{up}(i)\left(1 - \frac{\gamma}{|e(i)|}\right) e(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i), \tag{4-95}$$

Subtracting $\boldsymbol{a_o}$ from the last equation yields

$$\begin{aligned} \boldsymbol{v}(i+1) =& \boldsymbol{v}(i) + P_{up}(i)\left(1 - \frac{\gamma}{|e(i)|}\right) e(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i) \\ =& \boldsymbol{v}(i) + P_{up}(i)\,e(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i) - \gamma P_{up}(i)\,\mathrm{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i). \end{aligned} \tag{4-96}$$

Replacing (4-94) in the equation above we obtain

$$\begin{aligned} \boldsymbol{v}(i+1) =& \boldsymbol{v}(i) + P_{up}(i)\left(e_o(i) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{v}(i)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i) \\ & - \gamma P_{up}(i)\,\mathrm{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i) \\ =& \boldsymbol{v}(i) + P_{up}(i)\,e_o(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i) - P_{up}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{v}(i) \\ & - \gamma P_{up}(i)\,\mathrm{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i). \end{aligned} \tag{4-97}$$

Post-multiplying equation (4-97) by its transpose and taking the expected value leads to:

$$
\begin{aligned}
\mathbf{C}_{\boldsymbol{v}}(i+1) =& \mathbf{C}_{\boldsymbol{v}}(i) + P_{up}(i)\,\mathbb{E}\left[e_o(i)\,\boldsymbol{v}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
&- P_{up}(i)\,\mathbb{E}\left[\boldsymbol{v}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
&- P_{up}(i)\,\gamma\mathbb{E}\left[\operatorname{sgn}(e(i))\,\boldsymbol{v}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] + P_{up}(i)\,\mathbb{E}\left[e_o(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\right] \\
&+ P_{up}^2(i)\,\mathbb{E}\left[e_o^2(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
&- P_{up}^2(i)\,\mathbb{E}\left[e_o(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
&- 2P_{up}^2(i)\,\gamma\mathbb{E}\left[e_o(i)\operatorname{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
&- P_{up}(i)\,\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\right] \\
&- P_{up}^2(i)\,\mathbb{E}\left[e_o(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
&+ P_{up}^2(i)\,\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
&+ P_{up}^2(i)\,\gamma\mathbb{E}\left[\operatorname{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
&- P_{up}(i)\,\gamma\mathbb{E}\left[\operatorname{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\right] \\
&+ P_{up}^2(i)\,\gamma\mathbb{E}\left[\operatorname{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
&+ P_{up}^2(i)\,\gamma^2\mathbb{E}\left[\operatorname{sgn}^2(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right]. \qquad (4\text{-}98)
\end{aligned}
$$

Assuming that the inputs and the coefficients are statistically independent, then the following expected values are reduced to

$$\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\right] = \mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}(i), \qquad (4\text{-}99)$$

$$\mathbb{E}\left[\boldsymbol{v}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] = \mathbf{C}_{\boldsymbol{v}}(i)\,\mathbf{R}_{kk}. \qquad (4\text{-}100)$$

Let us also suppose that the optimum error is independent from the kernelized inputs. This assumption lead us to:

$$
\begin{aligned}
\mathbb{E}\left[e_o^2(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \approx& \mathbb{E}\left[e_o^2(i)\right]\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
\approx& J_{min}\mathbf{R}_{kk}, \qquad (4\text{-}101)
\end{aligned}
$$

and

$$
\begin{aligned}
\mathbb{E}\left[\operatorname{sgn}^2(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \approx& \mathbb{E}\left[\operatorname{sgn}^2(e(i))\right]\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}(i)\right] \\
\approx& \mathbf{R}_{kk}. \qquad (4\text{-}102)
\end{aligned}
$$

By the orthogonality principle, we obtain:

$$
\begin{aligned}
\mathbb{E}\left[e_o(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{v}^{\mathrm{T}}(i)\right] \approx& \mathbb{E}\left[e_o(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\right]\mathbb{E}\left[\boldsymbol{v}^{\mathrm{T}}(i)\right] \\
\approx& \mathbf{0}, \qquad (4\text{-}103)
\end{aligned}
$$

Let us also apply the orthogonality principle in the following expected value:

$$\begin{aligned}\mathbb{E}\left[e_o\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] &= \mathbb{E}\left[\boldsymbol{v}^{\mathrm{T}}\left(i\right)e_o\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\approx \mathbb{E}\left[\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right]\mathbb{E}\left[e_o\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\right]\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\approx \mathbf{0}. \end{aligned} \tag{4-104}$$

With the results of equations (4-99),(4-100), (4-101), (4-102), (4-103) and (4-104), equation (4-98) is reduced to:

$$\begin{aligned}\mathbf{C}_{\boldsymbol{v}}\left(i+1\right) &= \mathbf{C}_{\boldsymbol{v}}\left(i\right) - P_{up}\left(i\right)\mathbf{C}_{\boldsymbol{v}}\left(i\right)\mathbf{R}_{kk} - P_{up}\left(i\right)\gamma\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\quad + P_{up}^2\left(i\right)J_{min}\mathbf{R}_{kk} - 2P_{up}^2\left(i\right)\gamma\mathbb{E}\left[e_o\left(i\right)\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\quad - P_{up}\left(i\right)\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\left(i\right) + P_{up}^2\left(i\right)\mathbf{T}\left(i\right) \\ &\quad + P_{up}^2\left(i\right)\gamma\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\quad - P_{up}\left(i\right)\gamma\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \\ &\quad + P_{up}^2\left(i\right)\gamma\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\quad + P_{up}^2\left(i\right)\gamma^2\mathbf{R}_{kk}, \end{aligned} \tag{4-105}$$

where $\mathbf{T}\left(i\right) = \mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right]$. The remaining expected values of equation (4-105) can be computed using Price's theorem. For the ninth term, the expected value may be approximate as follows:

$$\begin{aligned}\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] &\approx \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \\ &\approx \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[\left(e_o\left(i\right) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \\ &\approx -\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \\ &\approx -\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\left(i\right). \end{aligned} \tag{4-106}$$

Calculating the third term of equation (4-105), we obtain

$$\begin{aligned}\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] &\approx \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\approx -\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{C}_{\boldsymbol{v}}\left(i\right)\mathbf{R}_{kk}. \end{aligned} \tag{4-107}$$

The sixth term of equation (4-105) is given by

$$\begin{aligned}\mathbb{E}\left[e_o\left(i\right)\operatorname{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \approx & \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e_o\left(i\right)e\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ \approx & \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e_o^2\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ & -\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e_o\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ \approx & \sqrt{\frac{2}{\pi\sigma_e^2}}J_{min}\mathbf{R}_{kk}. \end{aligned} \tag{4-108}$$

Finally, the eighth and the tenth terms can be computed by

$$\begin{aligned}\mathbb{E}\left[\operatorname{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] &\approx \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\approx -\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{T}\left(i\right). \end{aligned} \tag{4-109}$$

The results obtained in equations (4-106), (4-107), (4-108) and (4-109) turn equation (4-111) into:

$$\begin{aligned}\mathbf{C}_{\boldsymbol{v}}\left(i+1\right) = & \mathbf{C}_{\boldsymbol{v}}\left(i\right) - P_{up}\left(i\right)\mathbf{C}_{\boldsymbol{v}}\left(i\right)\mathbf{R}_{kk} + P_{up}\left(i\right)\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{C}_{\boldsymbol{v}}\left(i\right)\mathbf{R}_{kk} \\ & + P_{up}^2\left(i\right)J_{min}\mathbf{R}_{kk} - 2P_{up}^2\left(i\right)\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}J_{min}\mathbf{R}_{kk} \\ & - P_{up}\left(i\right)\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\left(i\right) + P_{up}^2\left(i\right)\mathbf{T}\left(i\right) \\ & - 2P_{up}^2\left(i\right)\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{T}\left(i\right) + P_{up}\left(i\right)\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\left(i\right) \\ & + P_{up}^2\left(i\right)\gamma^2\mathbf{R}_{kk}. \end{aligned} \tag{4-110}$$

Factorizing the common terms of the last equation, we get the following recursion for $\mathbf{C}_{\boldsymbol{v}}\left(i+1\right)$:

$$\begin{aligned}\mathbf{C}_{\boldsymbol{v}}\left(i+1\right) = & \mathbf{C}_{\boldsymbol{v}}\left(i\right) + P_{up}\left(i\right)\left(\gamma\sqrt{\frac{2}{\pi\sigma_e^2}} - 1\right)\left(\mathbf{C}_{\boldsymbol{v}}\left(i\right)\mathbf{R}_{kk} + \mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\right) \\ & + P_{up}^2\left(i\right)\left(1 - 2\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}\right)\left(J_{min}\mathbf{R}_{kk} + \mathbf{T}\left(i\right)\right) + P_{up}^2\left(i\right)\gamma^2\mathbf{R}_{kk}. \end{aligned} \tag{4-111}$$

The authors of [68] proved that the elements of $\mathbf{T}(i)$ are given by

$$\begin{aligned}[\mathbf{T}(i)]_{jj} =& \sum_{\substack{l=1 \\ l\neq j}}^{M} \left\{ 2\mu_2 [\mathbf{C}_{\boldsymbol{v}}(i)]_{jl} + \mu_3 [\mathbf{C}_{\boldsymbol{v}}(i)]_{ll} + \mu_4 \sum_{\substack{p=1 \\ p\neq\{j,l\}}}^{M} [\mathbf{C}_{\boldsymbol{v}}(i)]_{lp} \right\} \\ &+ \mu_1 [\mathbf{C}_{\boldsymbol{v}}(i)]_{jj}, \end{aligned} \tag{4-112}$$

for the main diagonal elements and

$$\begin{aligned}[\mathbf{T}(i)]_{jk} =& \mu_2 \left([\mathbf{C}_{\boldsymbol{v}}(i)]_{jj} + [\mathbf{C}_{\boldsymbol{v}}(i)]_{kk} \right) + 2\mu_3 [\mathbf{C}_{\boldsymbol{v}}(i)]_{jk} \\ &+ \sum_{\substack{l=1 \\ l\neq\{j,k\}}}^{M} \left\{ 2\mu_4 [\mathbf{C}_{\boldsymbol{v}}(i)]_{kl} + 2\mu_4 [\mathbf{C}_{\boldsymbol{v}}(i)]_{jl} + \mu_4 [\mathbf{C}_{\boldsymbol{v}}(i)]_{ll} \right. \\ &\left. + \mu_5 \sum_{\substack{p=1 \\ p\neq\{j,k,l\}}}^{M} [\mathbf{C}_{\boldsymbol{v}}(i)]_{lp} \right\}, \end{aligned} \tag{4-113}$$

for the off-diagonal entries, where μ_i is defined by

$$\mu_1 = \det\left\{ \mathbf{I}_2 - 4\mathbf{Q}_2\mathbf{R}_2/\nu^2 \right\}^{-\frac{1}{2}}, \tag{4-114}$$

$$\mu_2 = \det\left\{ \mathbf{I}_3 - \mathbf{Q}_{3'}\mathbf{R}_3/\nu^2 \right\}^{-\frac{1}{2}}, \tag{4-115}$$

$$\mu_3 = \det\left\{ \mathbf{I}_3 - 2\mathbf{Q}_{3'}\mathbf{R}_3/\nu^2 \right\}^{-\frac{1}{2}}, \tag{4-116}$$

$$\mu_4 = \det\left\{ \mathbf{I}_4 - 2\mathbf{Q}_4\mathbf{R}_4/\nu^2 \right\}^{-\frac{1}{2}}, \tag{4-117}$$

$$\mu_5 = \det\left\{ \mathbf{I}_5 - 2\mathbf{Q}_5\mathbf{R}_5/\nu^2 \right\}^{-\frac{1}{2}}, \tag{4-118}$$

and the matrices $\mathbf{Q}_i$ are defined by

$$\mathbf{Q}_{3'} = \begin{bmatrix} 4\mathbf{I} & -3\mathbf{I} & -\mathbf{I} \\ -3\mathbf{I} & 3\mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{4-119}$$

$$\mathbf{Q}_4 = \begin{bmatrix} 4\mathbf{I} & -2\mathbf{I} & -\mathbf{I} & -\mathbf{I} \\ -2\mathbf{I} & 2\mathbf{I} & \mathbf{0} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{4-120}$$

$$\mathbf{Q}_5 = \begin{bmatrix} 4\mathbf{I} & -\mathbf{I} & -\mathbf{I} & -\mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{4-121}$$

Replacing equations (4-112) and (4-113) into equation (4-111) leads us to a recursive expression for the entries of the autocorrelation matrix $\mathbf{C}_{\boldsymbol{v}}(i)$:

$$\begin{aligned} [\mathbf{C}_{\boldsymbol{v}}(i+1)]_{jj} = & \left(1 + 2P_{up}(i)\, ar_{md} + P_{up}^2(i)\, b\mu_1\right) [\mathbf{C}_{\boldsymbol{v}}(i)]_{jj} \\ & + P_{up}^2 b\mu_3 \sum_{\substack{l=1 \\ l\neq j}}^{M} [\mathbf{C}_{\boldsymbol{v}}(i)]_{ll} \\ & + \left(2P_{up}^2(i)\, \mu_2 b + 2P_{up}(i)\, ar_{od}\right) \sum_{\substack{l=1 \\ l\neq j}}^{M} [\mathbf{C}_{\boldsymbol{v}}(i)]_{jl} \\ & + P_{up}^2(i)\, \mu_2 b\mu_4 \sum_{\substack{l=1 \\ l\neq j}}^{M} \sum_{\substack{p=1 \\ p\neq\{j,l\}}}^{M} [\mathbf{C}_{\boldsymbol{v}}(i)]_{lp} \\ & + \left(P_{up}^2(i)\, bJ_{min} + P_{up}^2(i)\, \gamma^2\right) r_{md}, \end{aligned} \tag{4-122}$$

and for $j \neq k$

$$\begin{aligned} [\mathbf{C}_{\boldsymbol{v}}(i+1)]_{jk} = & \left(1 + 2P_{up}(i)\, \alpha r_{md} + 2P_{up}^2(i)\, \beta\mu_3\right) [\mathbf{C}_{\boldsymbol{v}}(i)]_{jk} \\ & + P_{up}^2(i)\, \beta\mu_4 \sum_{\substack{l=1 \\ l\neq\{j,k\}}}^{M} [\mathbf{C}_{\boldsymbol{v}}(i)]_{ll} + \Bigg(P_{up}^2(i)\, \beta\mu_2 \\ & + P_{up}(i)\, \alpha r_{od}\Bigg) \left([\mathbf{C}_{\boldsymbol{v}}(i)]_{jj} + [\mathbf{C}_{\boldsymbol{v}}(i)]_{kk}\right) \\ & + \left(2P_{up}^2(i)\, \beta\mu_4 + P_{up}(i)\, \alpha r_{od}\right) \sum_{\substack{l=1 \\ l\neq\{j,k\}}}^{M} \Bigg([\mathbf{C}_{\boldsymbol{v}}(i)]_{il} \\ & + [\mathbf{C}_{\boldsymbol{v}}(i)]_{jl}\Bigg) + P_{up}^2(i)\, \beta\mu_5 \sum_{\substack{l=1 \\ l\neq\{j,k\}}}^{M} \sum_{\substack{p=1 \\ p\neq\{j,k,l\}}}^{M} [\mathbf{C}_{\boldsymbol{v}}(i)]_{lp} \\ & + \left(P_{up}^2(i)\, \beta J_{min} + P_{up}^2(i)\, \gamma^2\right) r_{md}, \end{aligned} \tag{4-123}$$

where

$$\alpha = \gamma\sqrt{\frac{2}{\pi\sigma_e^2}} - 1, \tag{4-124}$$

$$\beta = 1 - 2\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}. \tag{4-125}$$

4.5 Simulations

In this section we perform several simulation experiments to test the algorithms proposed and compare them with the conventional algorithms. For this purpose, a time series prediction task was chosen. Two different time series were selected to carry out the experiments and evaluate the important characteristics of the algorithms proposed. Finally, we also corroborate the statistical analysis previously developed with simulations in a system identification environment

4.5.1 Time Series Prediction

The first time series chosen for the experiments is the Mackey-Glass time series, which is generated from a first-order non-linear differential equation described in [71] and given by

$$\frac{dx(t)}{dt} = -bx(t) + \frac{ax(t-\tau)}{1 + x(t-\tau)^n}, \tag{4-126}$$

where a, b, n and τ are real scalars. This equation was used to describe physiological control systems such as oxygen, glucose, blood cells and blood pressure in various organs. The time series displays characteristics of periodic and chaotic dynamics. The lag τ represents the time between the sensing of the value of the variable under control and the appropriate response. For this experiment we set b to 0.1, a to 0.2, n to 10 and τ to 30. The time series was generated with a sampling period of 6 seconds. Figure 4.1 shows the time series.

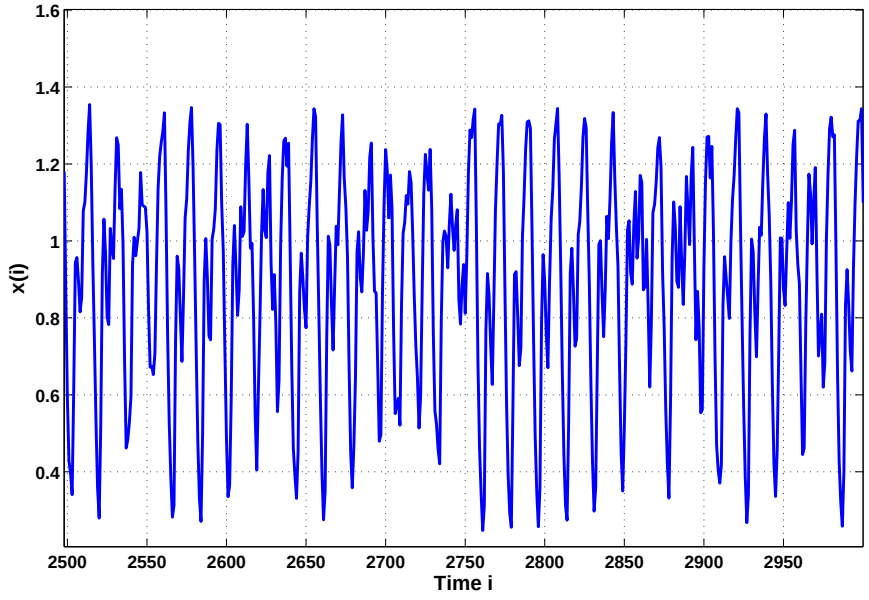

Figure 4.1: Mackey-Glass Time Series

The second time series is composed by a laser generated data. This data was collected in Germany and was recorded from a Far-Infrared-Laser in a chaotic state. The basic laser setup can be found in [72]. The intensity data were recorded by a LeCroy oscilloscope. The data are cross-cut through periodic to chaotic intensity pulsations of the laser as show in Figure 4.2. Chaotic pulsations more or less follow the theoretical Lorenz model [73] of a two level system.

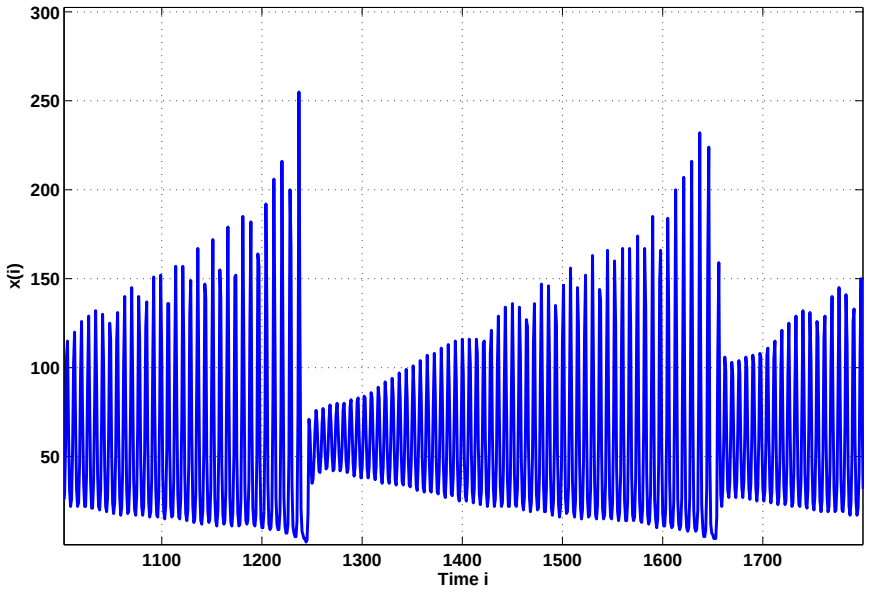

Figure 4.2: Laser Generated Time Series

In the first experiment we analyze the performance of five linear adaptive filters. We used both time series described before to perform the tests. First we separate the data into two sets, one for training and the other for testing [13]. A total of 3500 sample inputs were used for training and the prediction was performed over 100 test samples. The time-window was set to seven and the prediction horizon to one, so that the last seven inputs of the time series were used to predict the value one step ahead. Additionally both time series were corrupted by additive Gaussian noise with zero mean and standard deviation equal to 0.04. The step-size was set to 0.05 for all algorithms. For the AP and SM-AP algorithms, K was set to 7, i.e., so both algorithms used the last seven input samples as a single input for each prediction. The learning curves obtained for each time series are presented in Figure 4.3 and Figure 4.4, respectively. The fastest convergence speed is achieved by the SM-AP algorithm. We also see that the final mean square error is around $10^{-1.7}$ for both cases.

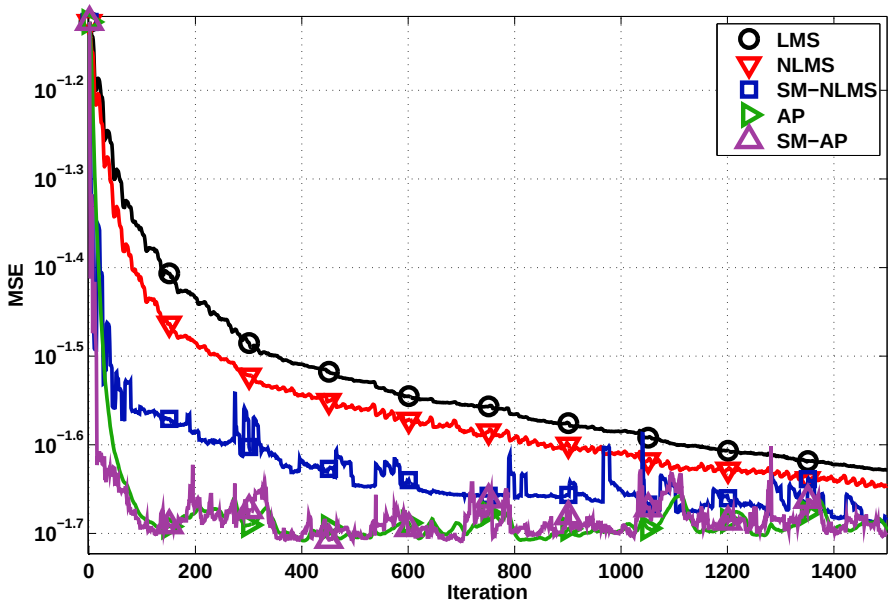

Figure 4.3: Linear adaptive filters learning curves: Mackey-Glass time series

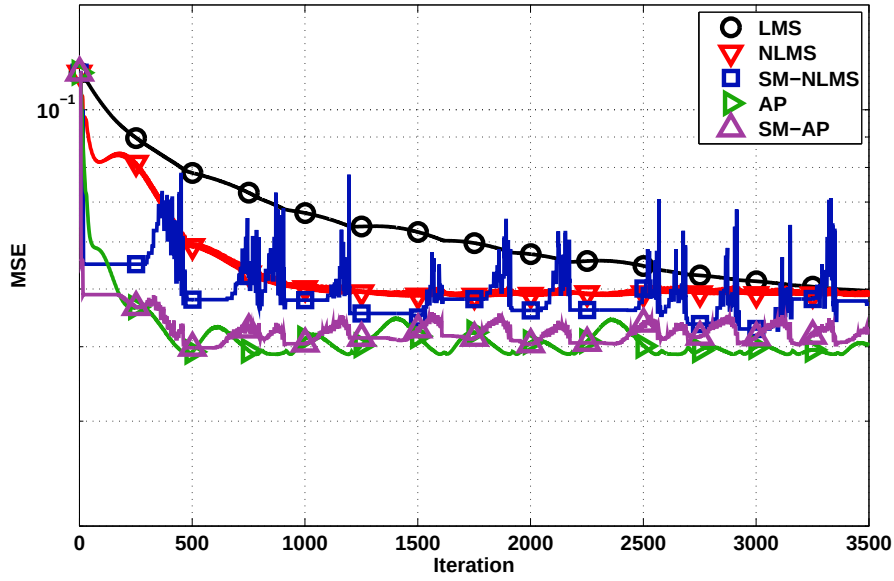

Figure 4.4: Linear adaptive filters learning curves: Laser generated time series.

For the next experiment we consider the performance of the algorithms proposed, the SM-NKLMS and the SM-KAP. We compare the results with the KLMS algorithm and the KAP algorithm. The Gaussian kernel was chosen to perform the experiments due its capability to model any input output mapping. After several tests, the bandwidth of the kernel was set to one for both algorithms. Once again the step-size was set to 0.05. The error bound

for both set-membership algorithms was set to $\sqrt{5}\sigma$. Finally, K was set to seven for the algorithms that reuse data. The results obtained are presented in Figures 4.5 and 4.6. The kernel-based algorithms achieve a steady-state MSE lower than 10^{-2}, so it is clear that the algorithms based on kernel functions outperform their linear counterparts.

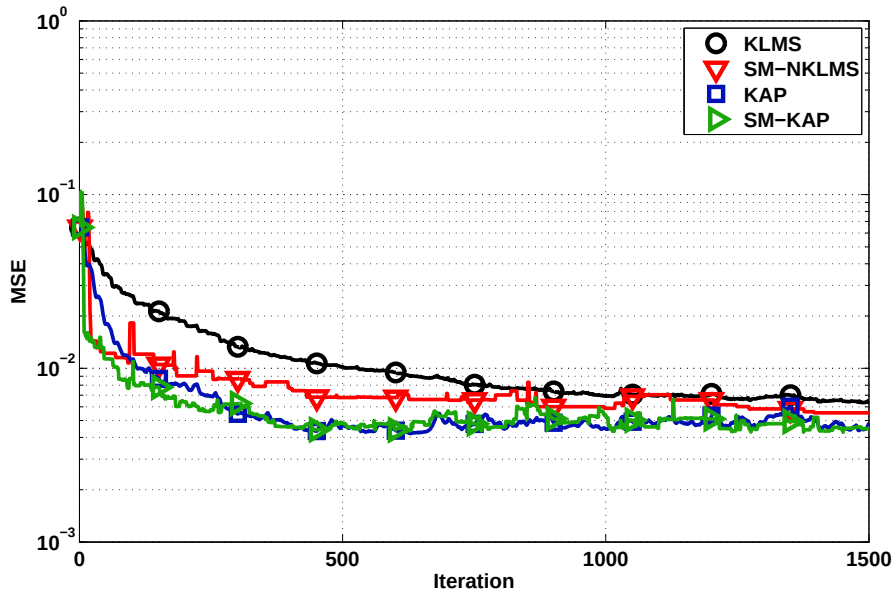

Figure 4.5: Kernel Algorithms Learning Curves: Mackey-Glass time series.

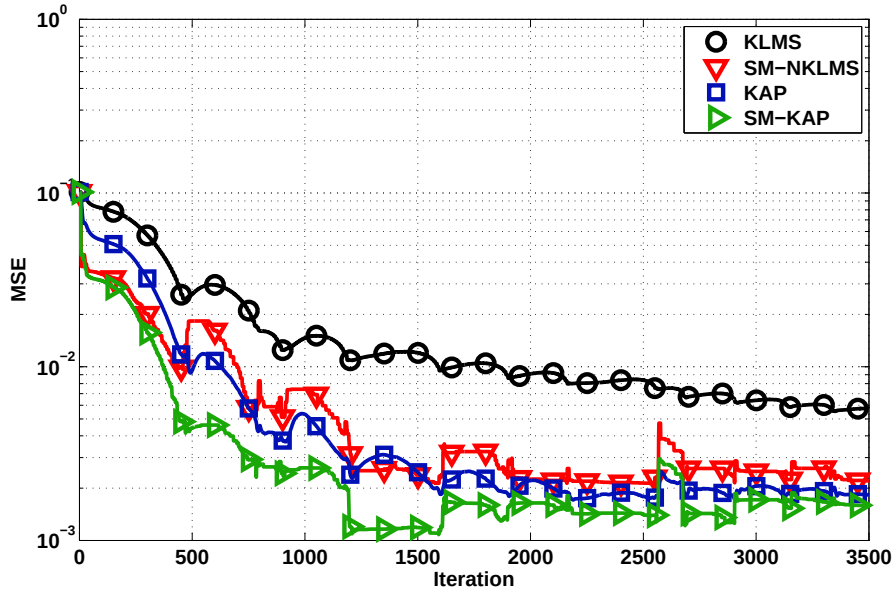

Figure 4.6: Kernel Algorithms Learning Curves: Laser generated time series.

Tables 4.1 and 4.2 summarizes the mean square error obtained by all the algorithms tested in the Makey-Glass time series and the laser data respectively, after 3500 iterations. The estimated output for the Mackey-Glass time series obtained with the SM-NKLMS algorithm is presented in Figure 4.7. Additionally, the estimated output for the laser generated time series obtained with SM-KAP algorithm is shown in Figure 4.8.

Table 4.1: Performance comparision on Mackey-Glass time series prediction

Algorithm	**Test Mean Square Error**	**Standard Deviation**
LMS	0.0233800	+/-0.00027135
NLMS	0.0223300	+/-0.00021383
SM-NLMS	0.0210950	+/-0.00035729
AP	0.0207910	+/-0.00178830
SM-AP	0.0201570	+/-0.00101700
KLMS	0.0028808	+/-0.00017820
SM-NKLMS	0.0022169	+/-0.00028659
KAP2	0.0019449	+/-0.00015231
SM-KAP	0.0018752	+/-0.00012415

Table 4.2: Performance comparision on laser generated time series prediction

Algorithm	**Test Mean Square Error**	**Standard Deviation**
LMS	0.0244290	+/-0.00035874
NLMS	0.0237260	+/-0.00101250
SM-NLMS	0.0230950	+/-0.00647190
AP	0.0215460	+/-0.00465890
SM-AP	0.0200020	+/-0.00154490
KLMS	0.0032337	+/-0.00067428
SM-NKLMS	0.0016275	+/-0.00054237
KAP2	0.0014548	+/-0.00030613
SM-KAP	0.0014071	+/-0.00019424

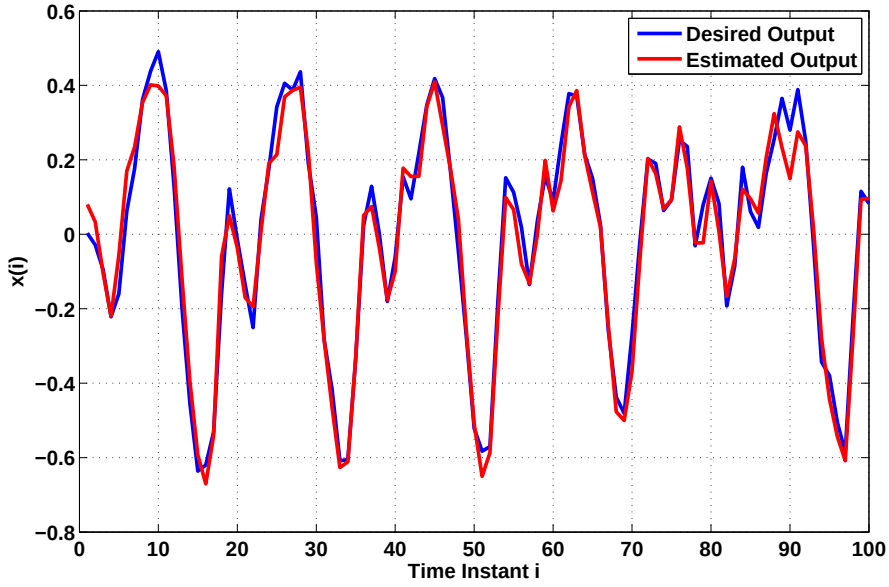

Figure 4.7: SM-NKLMS Algorithm Output: Mackey-Glass time series.

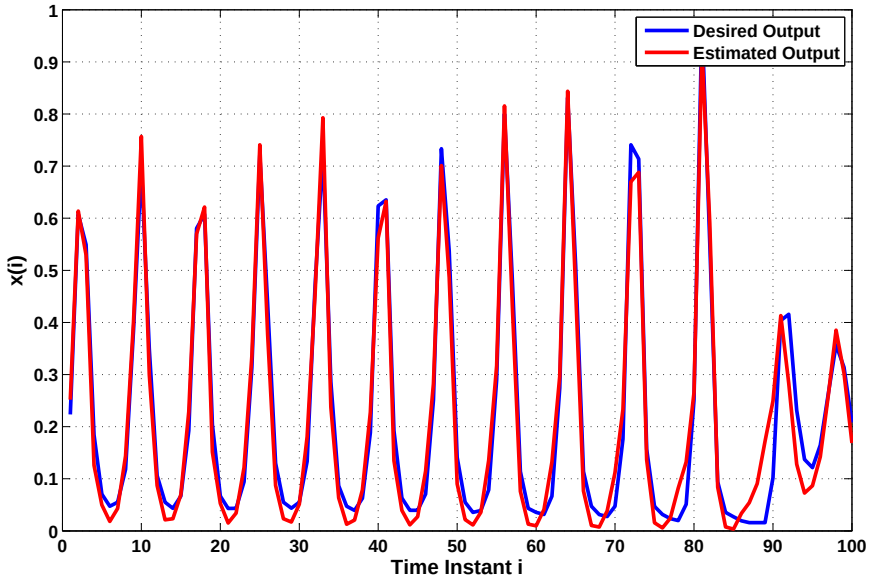

Figure 4.8: SM-KAP Algorithm Output: Laser generated time series.

For the next experiment, we analyse the length of the growing network created by the proposed SM-NKLMS algorithm using the laser generated time series. We also calculate the size of the growing network created by the KLMS algorithm with three different criteria, the coherence criterion (CC) [47, 48], the novelty criterion (NC) [45], and the surprise criterion (SC) [49].The results are showed in Figure 4.9. The proposed algorithm reduces significantly the size of the network created by the KLMS algorithm.

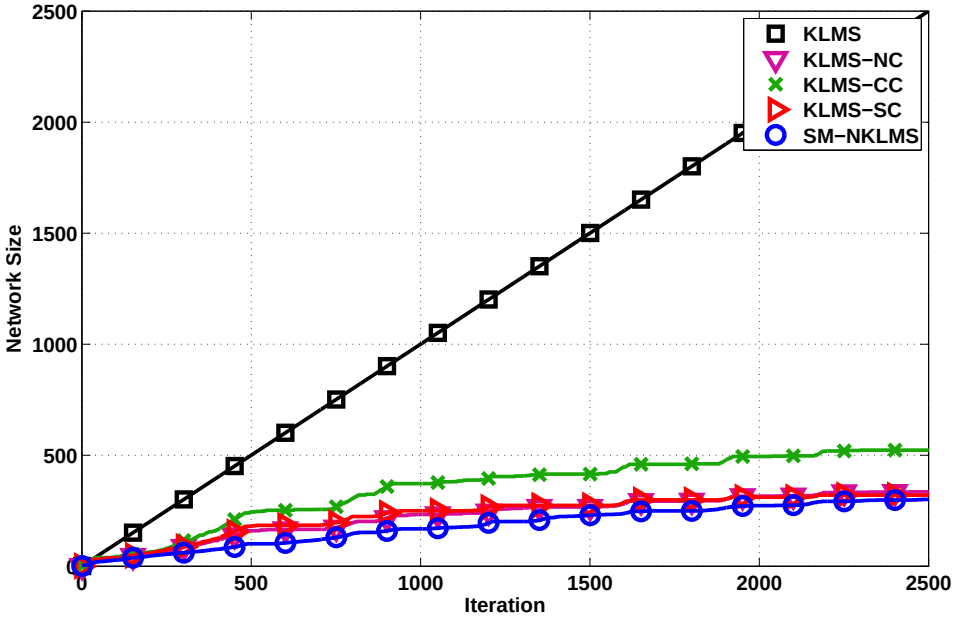

Figure 4.9: Dictionary size vs Iteration.

We also compare the performance of the SM-NKLMS with the performance obtained by the KLMS algorithm with different criteria. Figure 4.10 summarizes the results.

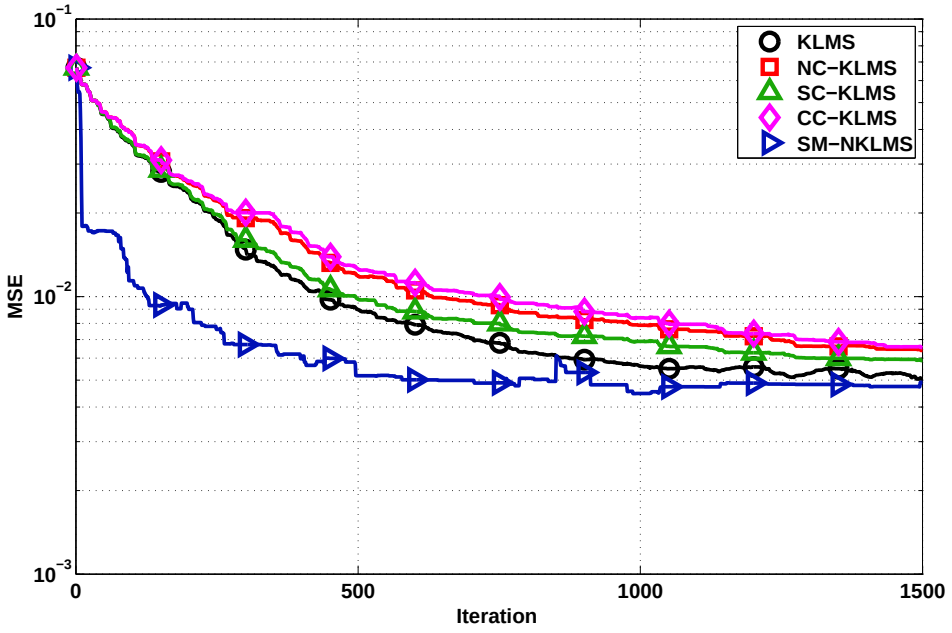

Figure 4.10: Performance comparision SM-NKLMS vs KLMS.

Table 4.3 compares the performance obtained for each algorithm. The KLMS algorithm with the surprise criterion obtains a performance sightly

worse than the KLMS. On the other hand, the SM-NKLMS shows a better convergence speed and also a better performance in terms of MSE. The size of the network after 2500 iterations for each criteria used is shown in Table 4.4. The parameters used by the different criteria may be tuned to achieve smaller network sizes, but the performances obtained degrades a lot.

Table 4.3: MSE obtained with different criteria

Algorithm	**Test Mean Square Error**	**Standard Deviation**
KLMS	0.0028564	+/-0.00048728
NC-KLMS	0.0040351	+/-0.00048059
SC-KLMS	0.0032697	+/-0.00032239
CC-KLMS	0.0041528	+/-0.00052414
SM-NKLMS	0.0024821	+/-0.00056300

Table 4.4: Network Size

Algorithm	**Network Size**
KLMS	2500
KLMS-NC	302
KLMS-CC	507
KLMS-SC	294
SM-NKLMS	278

Another experiment was set up to analyze the performance of the algorithms at different noise levels. Figure 4.11 shows the results obtained. In general, the kernel adaptive algorithms present a better robustness against noise compared to the linear adaptive algorithms.

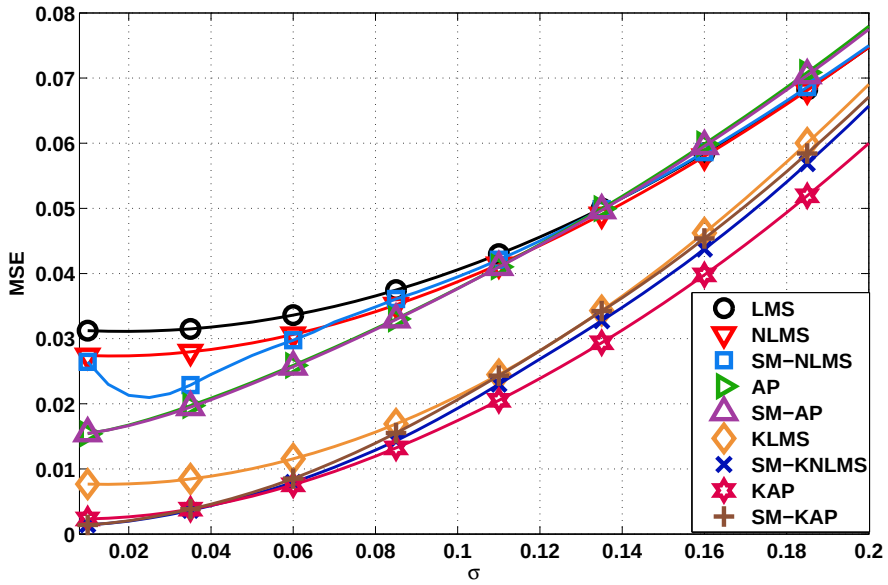

Figure 4.11: Performance of adaptive algorithms at different noise levels.

Finally, in order to see how the error bound affects the performance of the SM-NKLMS, we run the same simulation at different error bounds. From Figure 4.12 we see that the greater the value of the bound is, it achieves a stable behavior. However the MSE obtained with smaller values are better from those obtained with larger values. As we decrease the value of the error bound, we obtain noisier results.

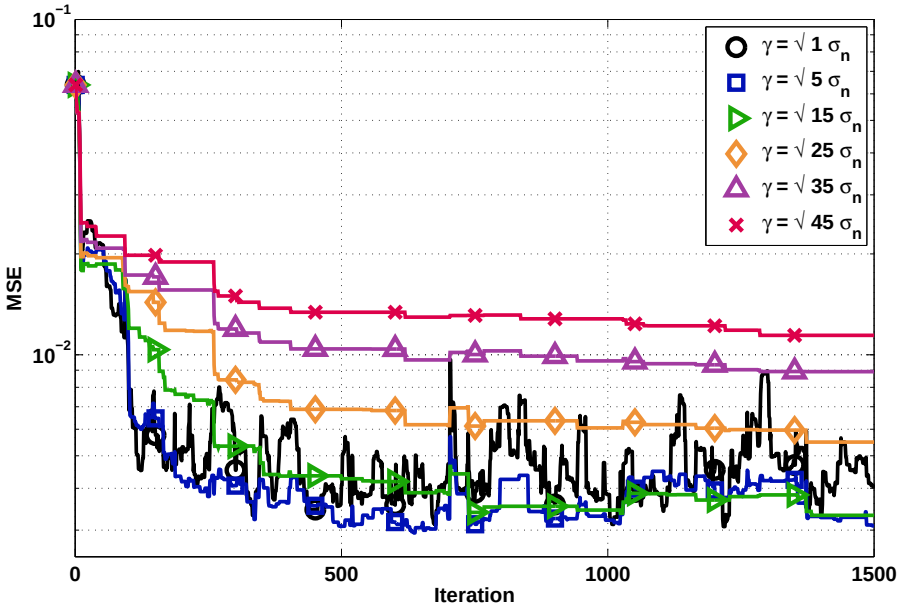

Figure 4.12: Performance of SM-NKLMS at different γ.

4.5.2
Transient behavior of the SM-NKLMS algorithm

The following system identification example is presented to verify the theory developed in Section 4.4.1. The nonlinear problem studied follows the recursion

$$d(i) = \frac{d(i-1)}{1+d(i-1)} + x^3(i-1). \tag{4-127}$$

The desired signal $d(i)$ was corrupted by additive white Gaussian noise with zero-mean and variance $\sigma_n^2 = 10^{-4}$. The input sequence $x(i)$ is independent and identically Gaussian distributed with variance $\sigma_x^2 = 0.1$. We also considered a fixed dictionary of length 20. At each iteration, the dictionary elements were updated so that the oldest element added is replaced. To compute the learning curve, a total of 500 simulations were averaged, each one with 3000 iterations. The bandwidth of the Gaussian kernel was set to 0.2 and 0.4. The results for two different values of γ are shown in Figures 4.13, 4.14, 4.15 and 4.16.

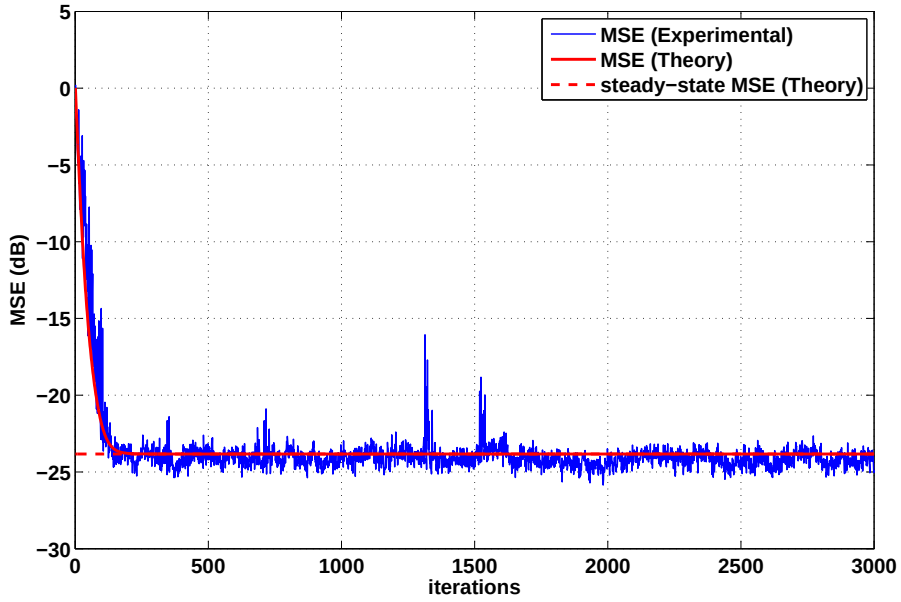

Figure 4.13: SM-NKLMS MSE behavior $\gamma = \sqrt{5}\sigma_n$, bandwidth= 0.2.

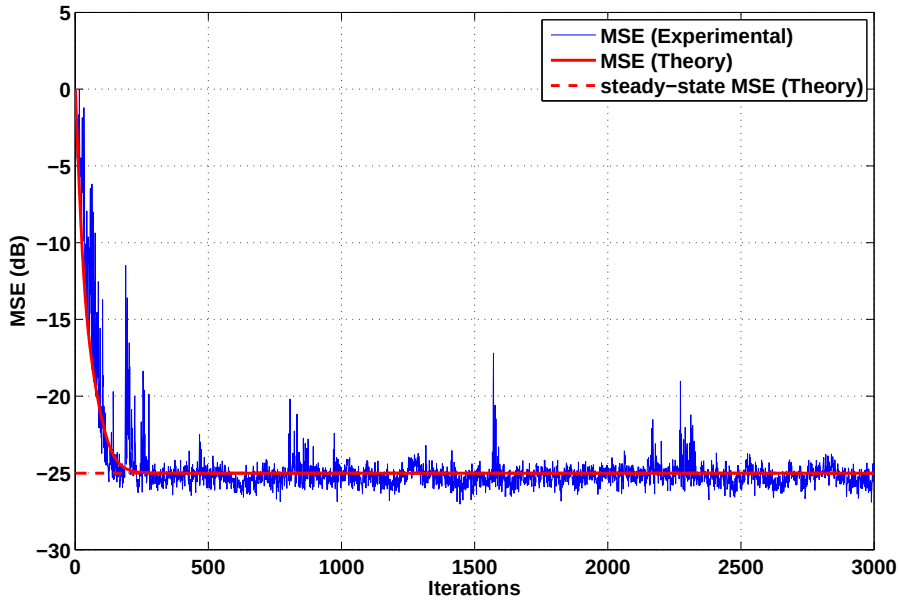

Figure 4.14: SM-NKLMS MSE behavior $\gamma = \sqrt{5}\sigma_n$, bandwidth= 0.4.

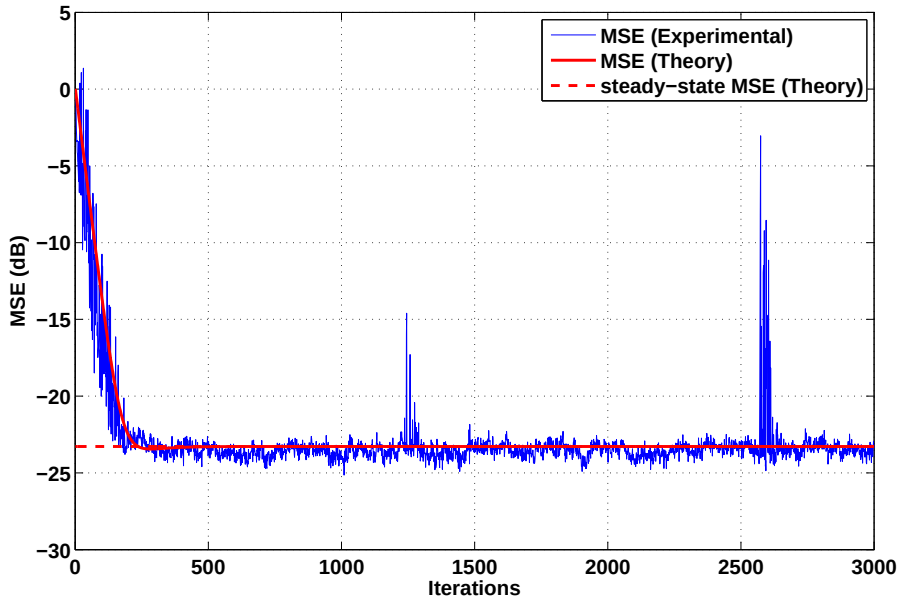

Figure 4.15: SM-NKLMS MSE behavior $\gamma = \sqrt{9}\sigma_n$, bandwidth= 0.2.

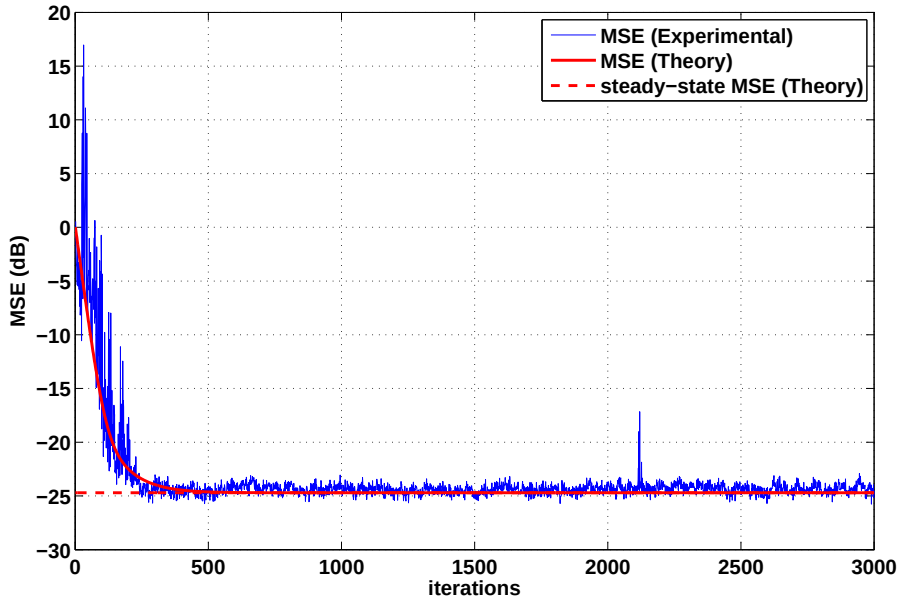

Figure 4.16: SM-NKLMS MSE behavior $\gamma = \sqrt{9}\sigma_n$, bandwidth= 0.4.

The lower steady-state MSE is obtained with a lower gamma. However, from the previous experiment we know that if we decrease the value of the error bound too much, we obtain noisier results. The mathematical model and expressions developed to describe the MSE of the proposed SM-NKLMS algorithm produce curves that match those obtained by simulation as evidenced by examples.

5
Conclusions and future work

In this dissertation, two kinds of data-selective adaptive algorithms have been derived and investigated, specifically linear algorithms and kernel-based algorithms. A sparse system identification and a time series prediction tasks were the application scenarios chosen to evaluate the proposed algorithms. The simulation results obtained showed that the proposed algorithms outperform conventional algorithms reported in previous works in terms of MSE and convergence speed.

In Chapter 3, a framework for data selective algorithms, that exploit the sparsity of the model by implementing a general penalty function into the cost function has been derived following a gradient descent approach. Using the established framework, three different algorithms with adjustable penalties have been presented. The proposed algorithms have a faster convergence speed than conventional adaptive algorithms used to exploit the sparsity of a system, making them an excellent choice for working in sparse environments.

We have also explored systems in the presence of correlated inputs. In such cases, the more correlated the input is, the slower the convergence speed and the higher the MSE achieved by the adaptive algorithms. However this condition has a lower impact in sparsity-aware algorithms than in conventional adaptive algorithms, constituting an advantage for the proposed algorithms.

In Chapter 4, we have shown that kernel-based adaptive algorithms are appropriate tools for time series prediction, where linear adaptive algorithms do not perform well. The proposed SM-NKLMS algorithm and SM-KAP algorithm, outperform the conventional KLMS and KAP algorithms in both, convergence speed and steady-state MSE level. The proposed SM-KAP present the best performance in terms of convergence rate and MSE value.

In Chapters 3 and 4 a statistical analysis to obtain a mathematical expression for the MSE at steady-state has been carried out for both kind of algorithms . The non-linear terms introduced by the analysed algorithms constitute the major difficulty to perform the analysis, requiring the use of different approximations. For both cases, the mathematical expression of the MSE matched the simulated result of the MSE, concluding that expressions found are a good approximation of the MSE value.

All algorithms proposed in this work present a substantial saving of computational resources. In Chapter 3 we conclude that only around 20%-25% of the data require an update. In Chapter 4, we found that the proposed algorithms not only performed less updates but they also limit the size of the dictionary, which represents a saving of memory resources. Because of all these facts, the proposed algorithms are ideal for systems with limited resources.

The proposed algorithms in this work can be considered for future work in other fields of application. In particular, they can be implemented for a channel equalization scenario and for a noise cancelling application. There are also other interesting fields where future work can be done. One example is the tracking analysis of the existing data-selective algorithms. In this research, we have considered only real data, but there are also many scenarios where complex data is collected. For these cases, the algorithms can be extended for considering complex values. Finally, we have restricted the proposed kernel-based algorithms to the use of a single kernel, however a multi-kernel approach could be an interesting topic of research.

Bibliography

[1] E. Fogel and Y.F. Huang. On the value of information in system identification-bounded noise case. *Automatica*, 18:229–238, March 1982.

[2] S. Gollamudi, S. Nagaraj, and Y. Kapoor, S.and Huang. Set-membership filtering and a set-membership normalized LMS algorithm with an adaptive step size. *IEEE Signal Processing Letters*, 5(5):111–114, May 1998.

[3] R. C. de Lamare and P. Diniz. Set-membership adaptive algorithms based on time-varying error bounds for CDMA interference suppression. *IEEE Transactions on Vehicular Technology*, 58(2):644 – 654, February 2009.

[4] T. Wang, R. C. de Lamare, and P.D. Mitchell. Low-complexity set-membership channel estimation for cooperative wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 60(6):2594 – 2607, July 2011.

[5] R. C. de Lamare and P. Diniz. Blind adaptive interference suppression based on set-membership constrained constant-modulus algorithms with dynamic bounds. *IEEE Transactions on Signal Processing*, 61(5):1288 – 1301, November 2013.

[6] L. Wang and R. C. de Lamare. Set-membership constrained conjugate gradient adaptive algorithm for beamforming. *IET Signal Processing*, 6(8):789 – 797, October 2012.

[7] L. Wang and R. C. de Lamare. Low-complexity constrained adaptive reduced-rank beamforming algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4):2114 – 2128, October 2013.

[8] P. Diniz. *Adaptive Filtering: Algorithms and Practical Implementation*. Springer, third edition, 2008.

[9] R. Pokharel, S. Seth, and J. Príncipe. Mixture kernel least mean square. *The 2013 International Joint Conference on Neural Networks*, 2013.

[10] M. Yukawa. Multikernel adaptive filtering. *IEEE Transactions on Signal Processing*, 60(9):4672 – 4682, August 2012.

[11] K. Slavakis and S. Theodoridis. Sliding window generalized kernel affine projection algorithm using projection mappings. *EURASIP Journal on Advances in Signal Processing*, 2008.

[12] K. Slavakis, S. Theodoridis, and I. Yamada. Online kernel-based classification using adaptive projection algorithms. *IEEE Transactions on Signal Processing*, 56(7):2781 – 2796, 2008.

[13] W. Liu, J. Príncipe, and S. Haykin. *Kernel Adaptive Filtering: A Comprehensive Introduction*. John Wiley & Sons, 2010.

[14] B. Widrow and S. Stearns. *Adaptive Signal Processing*. Prentice-Hall, 1985.

[15] J. Nagumo and A. Noda. A learning method for system identification. *IEEE Transactions on Automatic Control*, 12(3):282 – 287, 1967.

[16] K. Ozeki and T. Umeda. An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties. *Electronics and Communications in Japan*, 67-A(5):126–132, February 1984.

[17] S. Theodoridis. *Machine Learning A Bayesian and Optimization Perspective*. Academic press, 2015.

[18] R. C. de Lamare and R. Sampaio-Neto. Adaptive reduced-rank processing based on joint and iterative interpolation, decimation, and filtering. *IEEE Transactions on Signal Processing*, 57(7):2503 – 2514, March 2009.

[19] R. Fa, R. C. de Lamare, and L. Wang. Reduced-rank STAP schemes for airborne radar based on switched joint interpolation, decimation and filtering algorithm. *IEEE Transactions on Signal Processing*, 58(8):4182 – 4194, April 2010.

[20] Z. Yang, R. C. de Lamare, and X. Li. L_1-regularized STAP algorithms with a generalized sidelobe canceler architecture for airborne radar. *IEEE Transactions on Signal Processing*, 60(2):674 – 686, February 2012.

[21] Z. Yang, R. C. de Lamare, and X. Li. Sparsity-aware space-time adaptive processing algorithms with l_1-norm regularisation for airborne radar. *IET Signal Processing*, 6(5):413 – 423, July 2012.

[22] H. Ruan and R. C. de Lamare. Robust adaptive beamforming using a low-complexity shrinkage-based mismatch estimation algorithm. *IEEE Signal Processing Letters*, 21(1):60 – 64, January 2014.

[23] H. Ruan and R. C. de Lamare. Robust adaptive beamforming based on low-rank and cross-correlation techniques. *IEEE Transactions on Signal Processing*, 64(15):3919 – 3932, August 2016.

[24] T. G. Miller, S. Xu, R. C. de Lamare, and H. V. Poor. Distributed spectrum estimation based on alternating mixed discrete-continuous adaptation. *IEEE Signal Processing Letters*, 23(4):551 – 555, April 2016.

[25] D. L. Duttweiler. Proportionate normalized least-mean-squares adaptation in echo cancelers. *IEEE Transactions on Speech and Audio Processing*, 8(5):508 – 518, August 2000.

[26] J. Benesty and S. L. Gay. An improved PNLMS algorithm. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002.

[27] H. Deng and M. Doroslovacki. Proportionate adaptive algorithms for network echo cancellation. *IEEE Transactions on Signal Processing*, 54(5):1794 – 1803, April 2006.

[28] L. Liu, M Fukamoto, and S. Saiki. An improved mu-law proportionate NLMS algorithm. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008.

[29] F. de Sousa, O. Tobias, R. Seara, and D. R. Morgan. A PNLMS algorithm with individual activation factors. *IEEE Transactions on Signal Processing*, 58(4):2036 – 2047, April 2010.

[30] S. Werner, J. A. Apolinário Jr., P. S. R. Diniz, and T. I. Laakso. A set-membership approach to normalized proportionate adaptation algorithms. *13th European Signal Processing Conference*, 2005.

[31] C. Paleologu, S. Ciochina, and J. Benesty. An efficient proportionate affine projection algorithm for echo cancellation. *IEEE Signal Processing Letters*, 17(2):165 – 168, February 2010.

[32] O. Hoshuyama, R. A. Goubran, and A. Sugiyama. A generalized proportionate variable step-size algorithm for fast changing acoustic environments. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2004.

[33] S. Werner, J. A. Apolinário Jr., and P. S. R. Diniz. Set-membership proportionate affine projection algorithms. *EURASIP Journal on Audio, Speech and Music Processing*, 2007.

[34] Y. Chen, Y. Gu, and A. Hero. Sparse LMS for system identification. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3125–3128, April 2009.

[35] R. Meng, R. C. de Lamare, and V. H. Nascimento. Sparsity-aware affine projection adaptive algorithms for system identification. *Sensor Signal Processing for Defence*, 2011.

[36] X. Hong, J. Gao, and S. Chen. Zero attracting recursive least squares algorithms. *IEEE Transactions on Vehicular Technology*, 2016.

[37] E. M. Eksioglu and A. K. Tanc. RLS algorithm with convex regularization. *IEEE Signal Processing Letters*, 18(8):470 – 473, August 2011.

[38] M. Lima, T. Ferreira, W. Martins, and P. Diniz. Sparsity-aware data-selective adaptive filters. *IEEE Transactions on Signal Processing*, 62(17):4557–4572, August 2014.

[39] T. Ferreira, M. Lima, W. Martins, and P. Diniz. Modified sparsity-aware set-membership affine projection algorithm. *IEEE International Conference on Digital Signal Processing*, pages 833–837, July 2015.

[40] R. C. de Lamare and R. Sampaio-Neto. Sparsity-aware adaptive algorithms based on alternating optimization and shrinkage. *IEEE Signal Processing Letters*, 21(2):225–229, January 2014.

[41] B. K. Das, L. A. Azpicueta-Ruiz, M. Chakraborty, and J. Arenas-Garcia. A comparative study of two popular families of sparsity-aware adaptive filters. *4th International Workshop on Cognitive Information Processing*, 2014.

[42] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. and Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181 – 201, March 2001.

[43] S. Van Vaerenbergh, J. Via, and I. Santamaria. A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2006.

[44] S. Van Vaerenbergh, I. Santamaria, W. Liu, and J. Príncipe. Fixed-budget kernel recursive least-squares. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2010.

[45] J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3(3):213–225, 1991.

[46] Y. Engel, S. Mannor, and R. Meir. The kernel recursive least-squares algorithm. *IEEE Transactions on Signal Processing*, 52(8):2275–2285, August 2004.

[47] J. Pothin and C. Richard. Online learning with kernels. A new approach for sparsity control based on a coherence criterion. *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 241–245, 2006.

[48] C. Richard, J.C.M Bermudez, and P. Honeine. Online prediction of time series data with kernels. *IEEE Transactions on Signal Processing*, 57(3):1058–1067, February 2009.

[49] W. Liu and J. Príncipe. An information theoretic approach of designing sparse kernel adaptive filters. *IEEE Transactions on Neural Networks*, 20(12):1950 – 1961, November 2009.

[50] W. Liu, P. Pokharel, and J. Príncipe. The kernel least-mean-squares algorithm. *IEEE Transactions on Signal Processing*, 56(2):543–554, February 2008.

[51] W. Liu and J. Príncipe. Kernel affine projection algorithms. *EURASIP Journal on Advances in Signal Processing*, 2008, February 2008.

[52] P. Thomas. *A Study of Kernel- Based Adaptive Algorithms for Complex and Quaternion-Valued Data*. PhD thesis, Santa Clara University, October 2013.

[53] B. Schölkopf, R. Herbrich, and J. Smola. A generalized representer theorem. *14th Annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory*, pages 416–426, 2001.

[54] A. Sayed. *Adaptive Filters*. John Wiley & Sons, 2008.

[55] S. Haykin. *Adaptive Filter Theory*. Pearson, fifth edition, 2014.

[56] M.H. Hayes. *Statistical Digital Signal Processing and Modeling*. John Wiley & Sons, 1996.

[57] M. Lima and P. Diniz. On the steady-state MSE performance of the set-membership NLMS algorithm. *International Symposium on Wireless Communication Systems (ISWCS)*, 2010.

[58] G. Barrault, M. H. Costa, J. Bermudez, and A Lenzi. A new analitycal model for the NLMS algorithm. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005.

[59] M. H. Costa and J. Bermudez. An improved model for the normalized LMS algorithm with Gaussian inputs and large number of coefficients. *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002.

[60] J. Minkoff. Comment: on the unnecessary assumption of statistical independence between reference signal and filter weights in feedforward adaptive systems. *IEEE Transactions on Signal Processing*, 49(5):1109, 2001.

[61] N. R. Yousef and A. Sayed. A unified approach to the steady-state and tracking analyses of adaptive filters. *IEEE Transactions on Signal Processing*, 49(2):314 – 324, February 2001.

[62] M. Lima and P. Diniz. Steady-state analysis of the set membership affine projection algorithm. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3802–3805, March 2010.

[63] M. Lima and P. Diniz. Steady-state MSE performance of the set-membership affine projection algorithm. *Circuits, Systems, and Signal Processing*, 32(4):1811–1837, August 2013.

[64] R. Price. A useful theorem for nonlinear devices having Gaussian inputs. *IRE Transactions on Information Theory*, 4(2):69 – 72, June 1958.

[65] J. Chen, C. Richard, Y. Song, and D. Brie. Transient performance analysis of zero-attracting LMS. *IEEE Signal Processing Letters*, 23(12):1786 – 1790, 2016.

[66] R Coelho, V. H. Nascimento, R. Queiroz, J. Romano, and C Cavalcante, editors. *Signals and Images: Advances and Results in Speech, Estimation, Compression, Recognition, Filtering, and Processing*. CRC Press, 2015.

[67] J. Chen, W. Gao, C. Richard, and J. C. Bermudez. Convergence analysis of kernel LMS algorithm with pre-tuned dictionary. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[68] W. Parreira, J. C. Bermudez, C. Richard, and J Tourneret. Stochastic behavior analysis of the Gaussian kernel least-mean-square algorithm. *IEEE Transactions on Signal Processing*, 60(5):2208 – 2222, January 2012.

[69] W. Parreira, J. C. Bermudez, C. Richard, and J Tourneret. Steady-state behavior and design of the Gaussian KLMS algorithm. *European Signal Processing Conference (EUSIPCO)*, April 2011.

[70] J. Omura and T. Kailath. Some useful probability distributions. Technical Report 7050-6, Stanford University, 1965.

[71] M.C. Mackey and L. Glass. Oscillations and chaos in physiological control systems. *Science*, 197:287–289, 1977.

[72] U. Hubner, N.B. Abraham, and C.O. Weiss. Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared NH_3 laser. *Physical Review A*, 40(11):6354–6365, December 1989.

[73] E. Lorenz. Deterministic nonperiodic flow. *Journal of the atmospheric sciences*, 20:130–141, 1963.