Gaussian kernel least-mean-square : design, analysis and applications

Wei Gao

UNIVERSITÉ DE NICE-SOPHIA ANTIPOLIS - UFR Sciences
École Doctorale de Sciences Fondamentales et Appliquées

THESE

pour obtenir le titre de

Docteur en Sciences

de l'UNIVERSITE de Nice-Sophia Antipolis

Discipline : Sciences pour l'Ingénieur

présentée et soutenue par

Wei GAO

Gaussian kernel least-mean-square: design, analysis and applications

Thèse dirigée par **Cédric RICHARD**

soutenue le 9 décembre, 2015

Jury :

Reviewers:	Paul Honeine	Professor	-	Université de Rouen
	Jingdong Chen	Professor	-	Northwestern Polytechnical Univ.
Examinators:	André Ferrari	Professor	-	Université de Nice Sophia Antipolis
	Pierre-Olivier Amblard	DR CNRS	-	Grenoble INP
	Badong Chen	Professor	-	Xi'an Jiaotong University
	Jie Chen	Professor	-	Northwestern Polytechnical Univ.
Advisors:	Cédric Richard	Professor	-	Université de Nice Sophia Antipolis
	Jianguo Huang	Professor	-	Shanghai Jiao Tong Univ.

Gaussian kernel least-mean-square: design, analysis and applications

Abstract: The main objective of this thesis is to derive and analyze the Gaussian kernel least-mean-square (LMS) algorithm within three frameworks involving single and multiple kernels, real-valued and complex-valued, non-cooperative and cooperative distributed learning over networks. This work focuses on the stochastic behavior analysis of these kernel LMS algorithms in the mean and mean-square error sense. All the analyses are validated by numerical simulations.

First, we review the basic LMS algorithm, reproducing kernel Hilbert space (RKHS), framework and state-of-the-art kernel adaptive filtering algorithms.

Then, we study the convergence behavior of the Gaussian kernel LMS in the case where the statistics of the elements of the so-called dictionary only partially match the statistics of the input data. The theoretical analysis highlights the need for updating the dictionary in an online way, by eliminating obsolete elements and adding appropriate candidates. In order to automatically adjust the dictionary to the instantaneous stochastic properties of the input data, we introduced a modified kernel LMS algorithm based on forward-backward splitting to deal with ℓ_1-norm regularization. The stability of the proposed algorithm is then discussed.

After a review of two families of multikernel LMS algorithms, we focus on the convergence behavior of the multiple-input multikernel LMS algorithm. More generally, the characteristics of multikernel LMS algorithms are analyzed theoretically and confirmed by simulation results.

Next, the augmented complex kernel LMS algorithm is introduced based on the framework of complex multikernel adaptive filtering. Then, we analyze the convergence behavior of algorithm in the mean-square error sense.

Finally, in order to cope with the distributed estimation problems over networks, we derive functional diffusion strategies in RKHS. The proposed nonlinear diffusion adaptation with kernel LMS performs better than the non-cooperative kernel LMS algorithm. The stability of the algorithm in the mean sense is analyzed.

Keywords: convergence analysis, kernel least-mean-square, Gaussian kernel, multikernel, complex kernel, diffusion adaptation in RKHS

Kernel LMS à noyau Gaussien: conception, analyse et applications à divers contextes

Résumé: L'objectif principal de cette thèse est de décliner et d'analyser l'algorithme kernel-LMS à noyau Gaussien dans trois cadres différents: celui des noyaux uniques et multiples, à valeurs réelles et à valeurs complexes, dans un contexte d'apprentissage distributé et coopératif dans les réseaux de capteurs. Plus précisement, ce travail s'intéresse à l'analyse du comportement en moyenne et en erreur quadratique de cas différents types d'algorithmes LMS à noyau. Les modèles analytiques de convergence obtenus sont validés par des simulations numérique.

Tout d'abord, nous introduisons l'algorithme LMS, les espaces de Hilbert à noyau reproduisants, ainsi que les algorithmes de filtrage adaptatif à noyau existants.

Puis, nous étudions analytiquement le comportement de l'algorithme LMS à noyau Gaussien dans le cas où les statistiques des éléments du dictionnaire ne répondent que partiellement aux statistiques des données d'entrée. L'analyse théorique souligne la nécessité de mettre à jour le dictionnaire d'une manière en ligne, en éliminant les éléments obsolètes et en ajoutant les candidats appropriés. Afin d'adapter automatiquement le dictionnaire aux propriétés instantanées des données d'entrée, nous introduisons ensuite un algorithme LMS modifié à noyau basé sur une approche proximale. La stabilité de l'algorithme est également discutée.

Ensuite, nous introduisons deux types d'algorithmes LMS à noyaux multiples. Nous nous concentrons en particulier sur l'analyse de convergence de l'un d'eux. Puis, plus généralement, les caractéristiques des deux algorithmes LMS à noyaux multiples sont analysées théoriquement et confirmées par les simulations.

A la suite de cela, l'algorithme LMS à noyau complexe augmenté est présenté et ses performances analysées.

Enfin, dans un but d'estimation distribuée dans les réseaux de capteurs, nous proposons des stratégies de diffusion fonctionnelles dans les espaces de Hilbert à noyau reproduisant. La stabilité de cas l'algorithmes est étudiée.

Mots clés: analyse de convergence, kernel least-mean-square, noyau Gaussien, noyaux multiples, noyaux complexes, algorithmes de diffusion distributés, espaces de Hilbert à noyau reproduisant

Acknowledgments

In September 2012, I fortunately arrived at Nice Côte d'Azur and started to pursue my Ph.D study at University of Nice Sophia-Antipolis (UNS). Three years time fleeting, all the experiences of my life in Nice will become my precious memories and beneficial wealth in my future life. This thesis is a terminal point of amazing and unforgettable journey in which I have travelled with a lot of remarkable individuals who I really wish to acknowledge.

First and foremost, I would like to express my sincere gratitude to my advisor Prof. Cédric Richard not only for the continuous supports of my Ph.D study and his funding in my third year, but also for his patience, motivation, enthusiasm, and immense knowledge. He taught me what is the really excellent academic research work through numerous discussions and revisions of my manuscripts. I thank him deeply for the systematic guidance and great effort he put into training me in the signal processing community.

I am especially indebted to my co-advisor Prof. Jianguo Huang at Northwestern Polytechnical University (NPU), China. He encouraged and helped me with winning a scholarship to continue my study at UNS in France. I am grateful to him for all his encouragement and contribution to my thesis.

My sincere thanks also goes to Prof. Jose-Carlos M. Bermudez at Federal University of Santa Catarina, Brazil, for his time and detailed comments to my papers.

A very special thanks goes out to Prof. Jie Chen at NPU, China, for discussions and support in my research works and daily life whenever I was in need. I thank him also for his kindness and friendship all the time.

I would like to thank the other teachers in Lagrange Laboratory of UNS for their helps: Henri Lantéri, André Ferrari, Céline Theys, David Mary, Claude Aime, Jean-Pierre Folcher, Cédric Févotte and Rémi Flamary.

I must also acknowledge the financial support from the program of China Scholarship Council (CSC), and all the support from Lagrange Laboratory of University of Nice Sophia-Antipolis, UMR CNRS 7293, in particular Delphine Saissi and Jocelyne Bettini who provided me with all possible convenience for my work.

I would also like to thank all my colleagues and friends in Lagrange Lab: Dr. Liyong Liu, Dr. Silvia Paris, Dr. Raja Fazliza Suleiman, Dr. Rita Ammanouil, Dr. Roula Nassif and Rui Gao. I enjoyed the beautiful time with all of you together, and cherish our friendships forever. My sincere appreciations also goes to all the people who contribute to the completion of my thesis during the last three years.

Last but not the least, I take this opportunity to express the profound gratitude from my deep heart to my beloved parents and my elder sister for their love and continuous supporting both spiritually and materially throughout my life.

Contents

List of figures

List of tables

List of symbols and abbreviations

General notation

a, A	Lowercase and uppercase lightface letters denote scalars
$\boldsymbol{a}$	Lowercase boldface letters denote column vectors
$\boldsymbol{A}$	Uppercase boldface letters denote matrices
$\approx$	Approximated to
$\log_{10}$	Logarithm to the base 10
$\sum_{i=1}^{N}$	Summation
$a \sim F$	a follows distribution F
a^*	Complex conjugation of scalar a
$\lvert a \rvert$	Absolute value of scalar a
$\lVert \boldsymbol{a} \rVert$	Euclidean norm of a vector $\boldsymbol{a}$
$\lVert \boldsymbol{a} \rVert_p$	ℓ_p-norm of a vector $\boldsymbol{a}$
$E\{\boldsymbol{a}\}$	Expectation of $\boldsymbol{a}$
$\max(a, b)$	Larger value in a and b

Sets and spaces

$\{\boldsymbol{x} : P\}$	Set of $\boldsymbol{x}$ with property P
$\{i, \ldots, j\}$	Set of integers between i and j
$\mathbb{R}^M$	M-dimensional Euclidean space
$\mathbb{C}^M$	M-dimensional complex space
$\mathcal{H}$	Real functional Hilbert space
$\mathbb{H}$	Complex functional Hilbert space

Matrix and vector

$[a_1, a_2, \ldots, a_N]$	$(1 \times N)$ vector with components a_i, $i = 1, 2, \ldots, N$
$[\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_N]$	Matrix with column $\boldsymbol{a}_i$, $i = 1, 2, \ldots, N$

$[\boldsymbol{A}]_{ij}$	(i,j)th element of a matrix $\boldsymbol{A}$
$\mathbf{0}$	Vector of zeros
$\mathbf{1}$	Vector of ones
$\boldsymbol{O}$	Null matrix
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{A}^\top$	Transpose of a matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	Inverse of a matrix $\boldsymbol{A}$
$\boldsymbol{A}^H$	Complex-conjugate transposition for a matrix $\boldsymbol{A}$
$\det\{\boldsymbol{A}\}$	Determinant of a matrix $\boldsymbol{A}$
$\text{eig}_{\max}\{\boldsymbol{A}\}$	Maximum eigenvalue of a matrix $\boldsymbol{A}$
$\text{vec}\{\boldsymbol{A}\}$	Vectorizes matrix $\boldsymbol{A}$ and stacks its columns on top of each other
$\text{trace}\{\boldsymbol{A}\}$	Trace of a matrix $\boldsymbol{A}$
$\text{Diag}\{\boldsymbol{A}\}$	Vector composed by diagonal elements of a matrix $\boldsymbol{A}$

Abbreviations

i.i.d.	independent and identically distributed
s.t.	subject to
w.r.t.	with respect to
APA	Affine projection algorithm
EMSE	Excess mean squared error
FOBOS	Forward-backward splitting
KAF	Kernel adaptive filtering
KAP	Kernel affine projection
KLMS	Kernel least-mean-square
KRLS	Kernel recursive least-square
LMF	Least-mean-fourth
LMS	Least-mean-square
MSE	Mean squared error
NLMS	Normalized least-mean-square
PD	Positive-definite
RKHS	Reproducing kernel Hilbert space
SVM	Support vector machine

Chapter 1

Introduction

Contents

Learning is the activity or experience of acquiring new, or being taught and modifying, existing knowledge, skills, behaviors, or preferences and may involve synthesizing different types of information. With the advancement of human civilization, the ability to learn is possessed not only by human beings and animals but also by some artificial machines. Machine learning explores the construction and study of computer algorithms that improve automatically through experiences of learning from and making decision on data, and has been playing key role in artificial intelligence research. Depending on the context, machine learning tasks can be typically categorized into three fields as follows [Russell 1995, Bershad 2008, Duda 2012, Sutton 1998]: i) Supervised learning: The algorithm is trained to learn an underlying function that describes the relationship between inputs and desired outputs referred to as the training data. ii) Unsupervised learning: When outputs are not labelled, the unsupervised learning algorithm attempts to discover the hidden patterns in a stream of input. iii) Reinforcement learning: In reinforcement learning the agent is rewarded for good responses and punished for bad ones to sequentially form a strategy for operating in its problem space. These three machine learning categories have been widely employed to address a very broad range of problems in scientific and industrial fields.

According to the way in which the training data are generated and how they are provided, the learning models can be classified into batch (or offline) learning in which all the data have been collected at the beginning of learning, and online learning in which the learning algorithm gives an estimate of the output after receiving one sample of data at a time, before given the true value. In other words, online learning methods often update their current hypothesis in response to each new observation while their qualities of learning are evaluated by the total amount of estimated errors. Batch learning algorithms can generally present high-quality answer but require longer time and more resources. With the advent of streaming data sources, learning in some applications must often be performed to give medium-quality answer in real time, typically without a chance to revisit past entries. Therefore, learning in an online manner makes more sense in the era of data deluge.

1.1 Thesis context

Nowadays complex practical applications require more expressive hypothesis spaces than the limited computational capability of linear learning algorithms. Kernel representations offer an alternative solution that implicitly performs a nonlinear mapping from original input space into a high or infinite dimensional feature space to enhance the ability of existing linear learning algorithms dealing with nonlinear problems. Kernel methods offer powerful nonparametric modeling tools that have been successfully applied in support vector machines (SVM) [Scholkopf 2001], kernel principal component analysis [Schölkopf 1997], kernel ridge regression [Rosipal 2002, Cawley 2004], etc. This dissertation lies at the intersection of machine learning, in particular with kernel-based methods from supervised learning, and statistical signal processing that deals with the problem of finding a predictive function based on data.

Filtering is a fundamental statistical signal processing operation whose objective is to map the input signal to a desired output signal, in order to facilitate the extraction of desired information contained in the input signal. When dealing with signals whose statistical properties are unknown, or dynamic and depend on time, adaptive filters that automatically change their characteristics by adjusting some free parameters and structure are able to address these problems efficiently. The rationale of adaptive filters is to perform online updating of their parameters through a rather simple algorithm using the successively available information. Although the theory of adaptive filtering has been extensively studied for more than half century and has reached a relatively mature stage, and most of the research interests turned to seek specific implementations in diverse fields, this topic has been recently revitalized by the stimulation of novel requirements of nonlinear adaptive filtering with the development of theory of reproducing kernel Hilbert spaces (RKHS). During the last decade, adaptive filtering in RKHS has attracted substantial research interests due to its solid mathematical foundations, universal nonparametric modeling capacity, modest computational cost, etc.

Based on the principle of kernel representations, several state-of-the-art linear adaptive filters have been revisited to provide powerful kernel adaptive filtering algorithms operating in RKHS [Sayed 2003, Haykin 1991]. Typical kernel adaptive filtering algorithms include the kernel recursive least-squares algorithm (KRLS), which was initially proposed in [Engel 2004]. It can be viewed as a kernelized RLS algorithm, and is characterized by a fast convergence speed at the expense of a quadratic computational complexity. The sliding-window KRLS and extended KRLS algorithms were successively derived in [Steven 2006, Liu 2009b] to improve the tracking ability of the KRLS algorithm. The KRLS tracker algorithm was introduced in [Steven 2012], with ability to forget the past information using forgetting strategies. This allows the algorithm to track non-stationary input signals based on the idea of the exponentially-weighted KRLS algorithm [Liu 2010]. Next, a quantized KRLS algorithm (QKRLS) whose input space is quantized into smaller regions, was presented in [Chen 2013a]. The kernel affine projection (KAP) algorithm and, as a particular case, the kernel normalized least-mean-square (KNLMS) algorithm, were independently introduced in [Honeine 2007, Richard 2009, Slavakis 2008, Liu 2008b]. In addition, the kernel LMS (KLMS) algorithm as *de facto* standard was proposed and im-

proved in [Richard 2005, Liu 2008a, Chen 2012b]. It has attracted more attention because of its simplicity, superior tracking ability, robustness and linear computational complexity. Accordingly, this dissertation focuses on the analysis of the convergence behavior of KLMS algorithm.

An interesting alternative using projection-based methods in RKHS was also introduced in [Theodoridis 2011], leading to algorithms such as the sliding window generalized KAP algorithm [Slavakis 2008], the kernel adaptive projection subgradient method (KAPSM) [Slavakis 2013], and several interesting applications [Slavakis 2012]. A novel algorithm consisting of projection-along-subspace, selective update, and parallel projection, was recently developed to accelerate the convergence/tracking speed while reducing the computational complexity without serious degradation of performance [Takizawa 2015]. More recently, the nonlinear kernel Kalman filter was presented to improve tracking and prediction performance in [Zhu 2014]. Particle filtering based on kernel state-space model parametrized by reproducing kernel functions was proposed in [Tobar 2014b, Tobar 2015]. This unsupervised algorithm that performs inference on both parameters and hidden state through sequential Monte Carlo methods, was illustrated to outperform deterministic kernel adaptive filtering (KAF) algorithms.

In order to address the problem of kernel selection, multikernel learning has been extensively studied in the literature for classification and regression [Bach 2004, Sonnenburg 2006, Rakotomamonjy 2008, Kloft 2009, Martins 2011, Gonen 2011]. It has been shown that multikernel learning methods can enhance performance over monokernel methods since more flexibility and degree of freedom are provided by using multiple monokernels to handle nonlinearities. Investigations on multikernel learning have focused almost exclusively on batch processing, and only few efforts have been directed toward online processing. Multikernel LMS (MKLMS) algorithm was introduced independently in [Yukawa 2012, Yukawa 2013] and [Tobar 2012c, Tobar 2014a]. These works also proposed acceptance/rejection stages for kernel-based model selection in an online way. In [Tobar 2012c, Tobar 2014a], model selection is performed with novelty criterion. In [Yukawa 2012, Yukawa 2013], this step is accomplished with block ℓ_1-norm criteria. A variant of MKLMS algorithm with distinct dictionaries and different thresholds for model sparsification criterion was presented to further explore new degrees of freedom in [Ishida 2013]. The combination of the outputs of several monokernel LMS algorithms with different parameter settings was introduced to achieve improved performance in [Pokharel 2013, Gao 2014b].

All the kernel adaptive filtering algorithms mentioned above deal with real-valued data. Kernel-based adaptive filtering algorithms for complex data have also attracted attentions since they inherently ensure phase information processing [Bouboulis 2011, Bouboulis 2012]. This is of significance for applications in communication, radar, sonar, etc. A complexified kernel LMS algorithm and pure complex kernel LMS algorithm were introduced in [Bouboulis 2011]. Inspired by the linear augmented complex LMS (ACLMS) algorithm presented in [Mandic 2009, Kung 2009], its nonlinear counterpart augmented complex kernel normalized LMS (ACKNLMS) was described in [Bouboulis 2012, Tobar 2012a]. These works show that ACKNLMS algorithm can provide significantly improved performance compared with complex-valued algorithms, particularly for non-circular complex

data. As a recent trend in adaptive filtering with kernels, the quaternion KLMS algorithm was introduced as an extension of complex-valued KLMS algorithm in [Tobar 2013, Tobar 2014c, Ogunfunmi 2015, Paul 2015]. Finally, 2^m-dimensional Cayley-Dickson number system with $m \in \mathbb{Z}+$, was established for a translation of hypercomplex-valued linear system to real vector-valued data model [Mizoguchi 2013]. Through the algebraic translation the KAPSM was extended to possess the ability of processing hypercomplex number data in [Mizoguchi 2014].

Kernel adaptive filtering has recently become an appealing tool in many practical fields, including biomedical engineering [Camps-Valls 2007], remote sensing [Camps-Valls 2009, Chen 2013b, Chen 2014b, Dobigeon 2014], wind forecasting [Kuh 2009, Tobar 2012b] and automatic control [Xu 2007, Nguyen-Tuong 2012], to cite a few. In order to choose the appropriate algorithm along with its parameter set up subject to prescribed requirement of performance, theoretical performance and stability of algorithms are required to be thoroughly analyzed and investigated for both the transition and the steady-state stages.

1.2 Motivation

The main objective of this Ph.D. thesis is to derive theoretical models that characterize the convergence behavior of KLMS algorithm with Gaussian kernel under various assumptions and frameworks such as sliding window or pre-tuned dictionary, single kernel and multiple sub-kernels, real and complex kernels. The objective of this thesis is also to design novel kernel-based nonlinear filters such as distributed KLMS with diffusion strategies. Firstly, analytical models for convergence can be used to predict the behavior of each algorithm and used for designing kernel adaptive filters. Secondly, it becomes possible to accurately compare the performance of identical KLMS-type algorithms with different parameter settings such as step-size, kernel bandwidth, dictionary size. Finally, in order to meet various specific factors including rate of convergence, misadjustment, steady-state error performance, etc., theoretical analyses are powerful tools for practitioners to set KLMS parameters beforehand and offer a sufficient stability conditions. The substantial difficulties encountered in the theoretical convergence analysis of KLMS are now listed.

- The kernelized input sequence, instead of tap-input data of LMS algorithm, is no longer Gaussian-distributed.

- The dictionary has to be carefully deliberated as stochastic or preassigned as a part of filter parameters setting.

- There are challenges in dealing with the fourth-order moments.

- The impossibility of using the Gaussian moment-factoring theorem makes the convergence analysis of KLMS mathematically difficult.

1.3 Contributions

After a detailed study of the theoretical convergence behavior of KLMS algorithm in several situations, the main contributions of this thesis are the following:

1. Study of the convergence analysis of KLMS algorithm with single real-valued Gaussian kernel in the case where the statistics of the dictionary elements only partially match the statistics of the input data. This theoretical analysis highlights the need for updating the elements of dictionary in an online way. Stability analysis in the mean of KLMS with ℓ_1-norm regularization.

2. Theoretical analysis of the convergence behavior of multiple-input multikernel LMS algorithm in the mean and mean-square sense.

3. Analysis of the convergence behavior of the augmented complex Gaussian KLMS algorithm, proposed from the framework of complex multikernel adaptive filtering.

4. Derivation of distributed diffusion strategies in reproducing kernel Hilbert spaces over networks.

Several scientific publications were presented during the preparation of this dissertation:

Journal articles and International conference proceedings

1. Wei Gao, Jie Chen, Cédric Richard and Jianguo Huang. "Online dictionary learning for kernel LMS". *IEEE Transactions on Signal Processing*, vol. 62, no. 11, pages 2765–2777, June 2014.

2. Wei Gao, Jie Chen, Cédric Richard, Jose-Carlos M. Bermudez and Jianguo Huang. "Convergence analysis of the augmented complex KLMS algorithm with pre-tuned dictionary". *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 2006–2010, April 2015.

3. Wei Gao, Jie Chen, Cédric Richard and Jianguo Huang. "Diffusion adaptation over networks with kernel least-mean-square". *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pp. 1–4, December 2015.

4. Wei Gao, Cédric Richard, Jose-Carlos M. Bermudez and Jianguo Huang. "Convex combination of kernel adaptive filters". *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–5, September 2014.

5. Jie Chen, Wei Gao, Cédric Richard and Jose-Carlos M. Bermudez. "Convergence analysis of kernel LMS algorithm with pre-tuned dictionary". *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 7243–7247, May 2014.

6. Wei Gao, Jie Chen, Cédric Richard, Jianguo Huang and Rémi Flamary. "Kernel LMS algorithm with forward-backward splitting for dictionary learning". *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 5735–5739, May 2013.

7. Jianguo Huang, Wei Gao and Cédric Richard. "Multi-channel differencing adaptive noise cancellation based on kernel-based normalized least-mean-square algorithm". *MTS/IEEE International Conference on OCEANS*, pp.1–5, May 2012.

1.4 Organization of the thesis

The rest of this dissertation is organized as follows:

- Chapter 2 introduces linear adaptive filtering, in particular, the analysis of LMS algorithm. This chapter also includes a brief introduction to RKHS and their properties to be used in the following chapters. Existing kernel adaptive algorithms are presented at the end of Chapter 2.

- Chapter 3 justifies the necessity of updating kernel dictionary in an online way through the theoretical analysis of kernel LMS algorithm. Forward-backward splitting method is proposed to address this issue.

- Chapter 4 deals with the multikernel adaptive filtering, which is very attractive due to extra freedom degrees and flexibility. Special attention is given to the analysis of the multiple-input multikernel LMS.

- In Chapter 5, the basic principles of two families of complex monokernel LMS algorithms are reviewed. Then, augmented complex kernel LMS algorithm is introduced and analyzed.

- Chapter 6 addresses the topic of diffusion adaptation over networks for nonlinear distributed estimation. Simulation results show that the proposed cooperative strategy performs better than the non-cooperative usual monokernel LMS.

- Finally, a conclusion and possible future works are presented in Chapter 7.

Chapter 2

Linear and kernel adaptive filtering: a general overview

Contents

2.1 Introduction

The goal of this chapter is to briefly present the theoretical background materials of kernel adaptive filtering to make the understanding of the next chapters easier. This chapter consists of three main parts. In the first part, we review classic linear adaptive filtering algorithms in particular the LMS algorithm. The classical framework for analyzing of LMS convergence behavior is stated as an universal roadmap for ubiquitous analyses of KLMS-type algorithms in the sequel. The second part presents a brief introduction to the mathematical preliminaries for kernel-based learning algorithms. A brief overview of sparsification criteria for online dictionary learning and kernel adaptive filtering algorithms proposed is provided in the third part.

2.2 Linear adaptive filtering

2.2.1 Overview of linear adaptive filters

The aim of supervised linear adaptive filters is to automatically extract the information or identify the model from the noisy measurements of linear unknown system relying on error-correction criterion. The block diagram of a linear adaptive filter is shown in Figure 2.1. Distinct learning criteria result in different types of adaptive filters equipped with built-in mechanism of update. Commonly used adaptive filtering algorithms are summarized in the Table 2.1. These adaptive filtering algorithms have been successfully applied to address four basic problems including systems identification, inverse modeling, prediction and interference canceling [Haykin 1991]. The factors characterizing the performance of an adaptive filter include steady-state performance, transient behavior, tracking ability, robustness, etc.

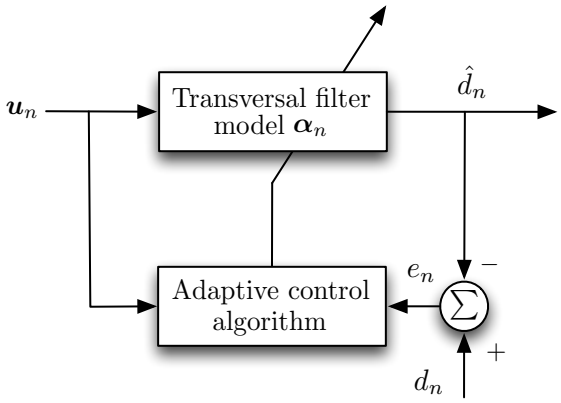


Figure 2.1: Block diagram of linear adaptive filter.

The well known least-mean-square (LMS) algorithm devised by Widrow and Hoff in 1959 [Widrow 1985] is an important member of the family of stochastic gradient-based algorithms, which iterate the weight of transversal filter in the direction of the negative gradient of the squared amplitude of the instantenous error signal with respect to each weight coefficient. By avoiding matrix inversion, LMS algorithm is simple and robust compared to other linear adaptive filtering algorithms. Consequently, it is considered as a benchmark when comparing the performance of different algorithms. It inspired kernel least-mean-square (KLMS) algorithm, and plays an important role in this dissertation. The next subsections briefly review basic but important principles and provide a convergence analysis of the LMS algorithm.

2.2.2 Wiener filter

Let the available input signal at time instant n be denoted by $\boldsymbol{u}_n = [u_n, u_{n-1}, \ldots, u_{n-L+1}]^\top$ with L the number of adjustable parameters in the model. This input signal is fed simultaneously into the system and model. The system output d_n is referred to as the desired

Table 2.1: Linear adaptive filtering algorithms.

Algorithms	Criteria	Recursions
LMS	$E\{(d_n - \boldsymbol{\alpha}^\top \boldsymbol{u}_n)^2\}$	$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, e_n \boldsymbol{u}_n$
NLMS	$E\{(d_n - \boldsymbol{\alpha}^\top \boldsymbol{u}_n)^2\}$	$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \frac{\eta}{\epsilon + \|\boldsymbol{u}_n\|^2} e_n \boldsymbol{u}_n, \quad \epsilon > 0$
sign-error LMS	$E\{\lvert d_n - \boldsymbol{\alpha}^\top \boldsymbol{u}_n\rvert\}$	$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, \mathrm{sign}\{e_n\}\, \boldsymbol{u}_n$
Leaky-LMS	$E\{(d_n - \boldsymbol{\alpha}^\top \boldsymbol{u}_n)^2\} + \lambda \|\boldsymbol{\alpha}\|^2$	$\boldsymbol{\alpha}_{n+1} = (1 - \eta\lambda)\boldsymbol{\alpha}_n + \eta\, e_n \boldsymbol{u}_n, \quad \lambda > 0$
LMF	$E\{(d_n - \boldsymbol{\alpha}^\top \boldsymbol{u}_n)^4\}$	$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, e_n \lvert e_n\rvert^2 \boldsymbol{u}_n$
APA	$\frac{1}{2}\|\boldsymbol{\alpha}_{n+1} - \boldsymbol{\alpha}_n\|^2$ s.t. $\boldsymbol{d}_n = \boldsymbol{U}_n^\top \boldsymbol{\alpha}_{n+1}$	$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, \boldsymbol{U}_n (\epsilon \boldsymbol{I} + \boldsymbol{U}_n^\top \boldsymbol{U}_n)^{-1} \boldsymbol{e}_n$ with $\boldsymbol{U}_n = [\boldsymbol{u}_n, \boldsymbol{u}_{n-1}, \ldots, \boldsymbol{u}_{n-p}]$
RLS	$\sum_{n=1}^{N} (d_n - \boldsymbol{\alpha}^\top \boldsymbol{u}_n)^2$	$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \boldsymbol{P}_n\, \boldsymbol{u}_n e_n$ $\boldsymbol{P}_n = \lambda^{-1}\big[\boldsymbol{P}_{n-1} - \frac{\boldsymbol{P}_{n-1}\boldsymbol{u}_n \boldsymbol{u}_n^\top \boldsymbol{P}_{n-1}}{\lambda + \boldsymbol{u}_n^\top \boldsymbol{P}_{n-1} \boldsymbol{u}_n}\big], \; 0 \ll \lambda < 1$

response or reference signal for linear adaptive filter to adjust the model parameters

$$d_n = \boldsymbol{\alpha}^\top \boldsymbol{u}_n + z_n \tag{2.1}$$

with $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \ldots, \alpha_L]^\top$ the weight vector. The input-desired data pairs $\{\boldsymbol{u}_n, d_n\}_{n=1}^N$ are assumed to be zero-mean stationary. The sequence z_n accounts for measurement noise and modeling error.

Consider the block diagram in Figure 2.2 built around a linear filter that aims at minimizing the mean-square error criterion

$$J(\boldsymbol{\alpha}) = E\left\{\left(d_n - \boldsymbol{\alpha}^\top \boldsymbol{u}_n\right)^2\right\} \tag{2.2}$$

whose solution $\boldsymbol{\alpha}$ satisfies the Wiener-Hopf equations

$$\boldsymbol{R}_{\boldsymbol{u}\boldsymbol{u}}\, \boldsymbol{\alpha} = \boldsymbol{p}_{\boldsymbol{u}d} \tag{2.3}$$

where $\boldsymbol{R}_{\boldsymbol{u}\boldsymbol{u}} = E\{\boldsymbol{u}_n \boldsymbol{u}_n^\top\}$ is the autocorrelation matrix of $\boldsymbol{u}_n$, and $\boldsymbol{p}_{\boldsymbol{u}d} = E\{d_n \boldsymbol{u}_n\}$ represents the cross-correlation vector between $\boldsymbol{u}_n$ and d_n. Assuming matrix $\boldsymbol{R}_{\boldsymbol{u}\boldsymbol{u}}$ is nonsingular, the optimal weight vector can be computed as

$$\boldsymbol{\alpha}_{\mathrm{opt}} = \boldsymbol{R}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\, \boldsymbol{p}_{\boldsymbol{u}d}. \tag{2.4}$$

This optimal solution is the so-called Wiener solution. Substituting the optimum weight vector $\boldsymbol{\alpha}_{\mathrm{opt}}$ into the mean-square-error (MSE) function (2.2), the corresponding minimum MSE is given by:

$$J_{\min} = E\left\{d_n^2\right\} - \boldsymbol{p}_{\boldsymbol{u}d}^\top\, \boldsymbol{R}_{\boldsymbol{u}\boldsymbol{u}}^{-1}\, \boldsymbol{p}_{\boldsymbol{u}d}. \tag{2.5}$$

An important property of the Wiener filter can be deduced if we analyze the gradient of the error surface at the optimal solution. Note that Wiener filter requires a priori information on the statistics of the input data, and is inadequate especially when dealing with intrinsic nonstationary signals in real-world applications.

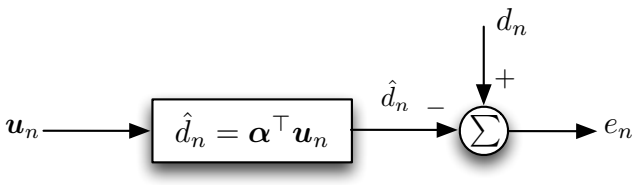


Figure 2.2: Block diagram of statistical filtering problem.

2.2.3 Least-mean-square algorithm

Let the mean-squared error or cost function $J(\boldsymbol{\alpha}_n)$ at time instant n be

$$J(\boldsymbol{\alpha}_n) = E\left\{e_n^2\right\} \tag{2.6}$$

where the estimation error e_n is denoted by

$$e_n = d_n - \boldsymbol{\alpha}_n^\top \boldsymbol{u}_n. \tag{2.7}$$

Instead of solving the Wiener-Hopf equations, the steep-descent method can be applied to minimize the cost function defined in (2.6), and iteratively calculate the weight vector by using a simple recursive relation

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \frac{1}{2}\,\eta\,[-\nabla J(\boldsymbol{\alpha}_n)] \tag{2.8}$$

where $\eta > 0$ is the step-size parameter or weighting constant. The factor $\frac{1}{2}$ is used for convenience. The gradient vector $\nabla J(\boldsymbol{\alpha}_n)$ at time instant n can be computed as follows:

$$\begin{aligned}\nabla J(\boldsymbol{\alpha}_n) &= \left[\frac{\partial J(\boldsymbol{\alpha}_n)}{\alpha_{n,1}}, \frac{\partial J(\boldsymbol{\alpha}_n)}{\alpha_{n,2}}, \ldots, \frac{\partial J(\boldsymbol{\alpha}_n)}{\alpha_{n,L}}\right]^\top \\ &= -2\,\boldsymbol{p}_{\boldsymbol{u}d} + 2\,\boldsymbol{R}_{\boldsymbol{u}\boldsymbol{u}}\boldsymbol{\alpha}_n.\end{aligned} \tag{2.9}$$

Accordingly, the simple recursive update of weight vector $\boldsymbol{\alpha}_{n+1}$ can be written as

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\,(\boldsymbol{p}_{\boldsymbol{u}d} - \boldsymbol{R}_{\boldsymbol{u}\boldsymbol{u}}\boldsymbol{\alpha}_n)\,. \tag{2.10}$$

In order to get a practical adaptive filtering algorithm, the intuitively simplest strategy is to consider the instantaneous estimates of $\boldsymbol{R}_{\boldsymbol{u}\boldsymbol{u}}$ and $\boldsymbol{p}_{\boldsymbol{u}d}$ defined by, respectively,

$$\hat{\boldsymbol{R}}_{\boldsymbol{u}\boldsymbol{u}} = \boldsymbol{u}_n\,\boldsymbol{u}_n^\top \tag{2.11}$$

and

$$\hat{\boldsymbol{p}}_{\boldsymbol{u}d} = d_n\,\boldsymbol{u}_n. \tag{2.12}$$

Correspondingly, the instantaneous gradient vector is given by

$$\hat{\nabla} J(\boldsymbol{\alpha}_n) = -2\,\hat{\boldsymbol{p}}_{\boldsymbol{u}d} + 2\,\hat{\boldsymbol{R}}_{\boldsymbol{u}\boldsymbol{u}}\boldsymbol{\alpha}_n. \tag{2.13}$$

Substituting the estimate of (2.9) into the steep-descent algorithm (2.10), we obtain the recursive equation

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, \boldsymbol{u}_n \left(d_n - \boldsymbol{\alpha}_n^\top \boldsymbol{u}_n \right). \tag{2.14}$$

Equivalently, the well known LMS algorithm is written more frequently as

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, e_n \boldsymbol{u}_n \tag{2.15}$$

with the instantaneous error denoted by $e_n = d_n - \boldsymbol{\alpha}_n^\top \boldsymbol{u}_n$. The computational requirement of the LMS algorithm for L-order filter are only $2(L+1)$ multiplications and $2L$ additions per iteration. In order to reduce the sensitivity of standard LMS to the magnitude of input signal, the normalized LMS (NLMS) algorithm was proposed by introducing a time-varying step-size that is inversely proportional to variance of the input, namely,

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \frac{\eta}{\epsilon + \|\boldsymbol{u}_n\|^2} e_n \boldsymbol{u}_n \tag{2.16}$$

with $\epsilon > 0$ a regularization parameter to avoid singularity. In the next subsection, we briefly review the main steps of an analysis of LMS convergence behavior.

2.2.4 LMS convergence analysis

Convergence analysis of the LMS algorithm is based on two criteria:

- Convergence in the mean of the weight vector $\boldsymbol{\alpha}_n$, that is,

$$\lim_{n\to\infty} E\{\boldsymbol{\alpha}_n\} = \boldsymbol{\alpha}_{\text{opt}}. \tag{2.17}$$

- Convergence in the mean-square sense of the weight-error vector $\boldsymbol{v}_n = \boldsymbol{\alpha}_n - \boldsymbol{\alpha}_{\text{opt}}$, that is,

$$\lim_{n\to\infty} E\{J_{\text{ms}}(\boldsymbol{\alpha}_n)\} = J_{\text{ms}}(\infty). \tag{2.18}$$

These two criteria allow us to determine two necessary conditions for convergence of LMS in the mean and the mean square sense, respectively. To make the convergence analysis of the LMS algorithm mathematically tractable, it is necessary to introduce some fundamental independence assumptions (IA) consisting of

1. The input data $\{\boldsymbol{u}_n\}_{n=1}^N$ are statistically independent vectors.

2. The input vector $\boldsymbol{u}_n$ is statistically independent of all previous samples of the desired signal $\{d_1, d_2, \ldots, d_{n-1}\}$.

3. The desired response d_n is only dependent on $\boldsymbol{u}_n$, but statistically independent of all its previous samples.

2.2.4.1 Mean weight vector behavior

Subtracting the optimal weight vector $\boldsymbol{\alpha}_{\text{opt}}$ from both sides of equation (2.14) and using the definition of $\boldsymbol{v}_n$, we obtain

$$\boldsymbol{v}_{n+1} = \big(\boldsymbol{I} - \eta\, \boldsymbol{u}_n \boldsymbol{u}_n^{\top}\big)\boldsymbol{v}_n + \eta\, e_{\text{opt}}(n)\, \boldsymbol{u}_n \tag{2.19}$$

where $e_{\text{opt}}(n) = d_n - \boldsymbol{\alpha}_{\text{opt}}^{\top}\boldsymbol{u}_n$ using Wiener solution. Taking the expected value of both sides of equation (2.19) and using the IA assumptions, which states that $\boldsymbol{v}_n$ and $\boldsymbol{R}_{\boldsymbol{uu}}$ are independent, yields

$$E\left\{\boldsymbol{v}_{n+1}\right\} = \big(\boldsymbol{I} - \eta\, \boldsymbol{R}_{\boldsymbol{uu}}\big) E\left\{\boldsymbol{v}_n\right\}. \tag{2.20}$$

Then the following theorem on the convergence behavior of $\boldsymbol{\alpha}_n$ can be stated as follows:

Theorem 2.2.1 *The weight vector of the LMS algorithm converges to the Wiener solution as n tends to infinity, namely, $\lim_{n\to\infty} E\{\boldsymbol{\alpha}_n\} = \boldsymbol{\alpha}_{\text{opt}}$, if and only if the step-size satisfied*

$$0 < \eta < \frac{2}{\text{eig}_{\max}\{\boldsymbol{R}_{\boldsymbol{uu}}\}}. \tag{2.21}$$

Under this condition, we say that the LMS algorithm is convergent in the mean.

2.2.4.2 Mean-square-error behavior

We shall now derive a recursive equation on the correlation matrix of the weight-error vector $\boldsymbol{v}_n$ defined by:

$$\boldsymbol{K}_n = E\{\boldsymbol{v}_n \boldsymbol{v}_n^{\top}\}. \tag{2.22}$$

Post-multiplying equation (2.19) by its transpose, and taking the expected value, yields (See [Haykin 1991] for detailed procedures)

$$\begin{aligned} \boldsymbol{K}_{n+1} &= \boldsymbol{K}_n - \eta\left(\boldsymbol{R}_{\boldsymbol{uu}}\boldsymbol{K}_n + \boldsymbol{K}_n\boldsymbol{R}_{\boldsymbol{uu}}\right) + \eta^2 \boldsymbol{R}_{\boldsymbol{uu}}\text{trace}\{\boldsymbol{R}_{\boldsymbol{uu}}\boldsymbol{K}_n\} \\ &\qquad + \eta^2 \boldsymbol{R}_{\boldsymbol{uu}}\boldsymbol{K}_n\boldsymbol{R}_{\boldsymbol{uu}} + \eta^2 J_{\min}\boldsymbol{R}_{\boldsymbol{uu}} \\ &= (\boldsymbol{I} - \eta\boldsymbol{R}_{\boldsymbol{uu}})\boldsymbol{K}_n(\boldsymbol{I} - \eta\boldsymbol{R}_{\boldsymbol{uu}}) + \eta^2 \boldsymbol{R}_{\boldsymbol{uu}}\text{trace}\{\boldsymbol{R}_{\boldsymbol{uu}}\boldsymbol{K}_n\} + \eta^2 J_{\min}\boldsymbol{R}_{\boldsymbol{uu}}. \end{aligned} \tag{2.23}$$

Consequently, the above recursive update equation for $\boldsymbol{K}_n$ provides us a tool for evaluating the transient behavior of the MSE of LMS algorithm

$$\begin{aligned} J_{\text{ms}}(\boldsymbol{\alpha}_n) &= E\left\{(d_n - \boldsymbol{\alpha}_n^{\top}\boldsymbol{u}_n)^2\right\} \\ &= J_{\min} + \text{trace}\{\boldsymbol{R}_{\boldsymbol{uu}}\boldsymbol{K}_n\}. \end{aligned} \tag{2.24}$$

The difference between the MSE $J_{\text{ms}}(\boldsymbol{\alpha}_n)$ and minimum MSE $J_{\min}$ is the second term on right hand side of (2.24). It is defined as the excess MSE (EMSE) denoted by:

$$\begin{aligned} J_{\text{ex}}(n) &= J_{\text{ms}}(\boldsymbol{\alpha}_n) - J_{\min} \\ &= \text{trace}\{\boldsymbol{R}_{\boldsymbol{uu}}\boldsymbol{K}_n\}. \end{aligned} \tag{2.25}$$

The transient EMSE $J_{\text{ex}}(n)$ can be calculated by using the recursive relation (2.23). Observe that the mean-square estimation error of LMS algorithm is determined by two components: the minimum MSE $J_{\min}$ and the EMSE $J_{\text{ex}}(n)$. When time $n \to \infty$ the steady-state EMSE $J_{\text{ex}}(n)$ can be expressed in terms of the eigenvalues λ_ℓ of correlation matrix $\boldsymbol{R_{uu}}$ [Haykin 1991]

$$J_{\text{ex}}(\infty) = J_{\min} \frac{\sum_{\ell=1}^{L} \eta\lambda_\ell/(2-\eta\lambda_\ell)}{1 - \sum_{\ell=1}^{L} \eta\lambda_\ell/(2-\eta\lambda_\ell)}. \tag{2.26}$$

2.3 Preliminaries on kernel-based methods

In this section, we provide some insights into the fundamental concepts and properties characterizing kernel functions instead of fully rigorous presentation with necessary proofs. Most of the material presented here can also be found with more details in several papers and textbooks [Aronszajn 1950, Schölkopf 2000, Shawe-Taylor 2004, Slavakis 2013].

2.3.1 Definition of kernel

Following the development of SVM, the kernel-based methods attracted considerable attention in the machine learning community. In order to study kernel adaptive filtering, we need first to introduce necessary definition related to kernels and some useful properties.

Definition 2.3.1 (Kernel function) *Let $\mathcal{U} \subset \mathbb{R}^n$ be a nonempty set. For all $\boldsymbol{u}, \boldsymbol{u}' \in \mathcal{U}$ a kernel is a function κ defined as*

$$\begin{aligned} \kappa : \mathcal{U} \times \mathcal{U} &\longrightarrow \mathbb{R} \\ (\boldsymbol{u}, \boldsymbol{u}') &\longmapsto \kappa(\boldsymbol{u}, \boldsymbol{u}'). \end{aligned} \tag{2.27}$$

The kernel function generally returns a real number characterizing the similarity of two inputs $\boldsymbol{u}$ and $\boldsymbol{u}'$. If given a non-unit norm kernel $\kappa(\boldsymbol{u}, \boldsymbol{u}')$ that corresponds to the feature mapping ϕ, its normalized kernel $\bar{\kappa}(\boldsymbol{u}, \boldsymbol{u}')$ can be expressed as

$$\bar{\kappa}(\boldsymbol{u}, \boldsymbol{u}') = \left\langle \frac{\phi(\boldsymbol{u})}{\|\phi(\boldsymbol{u})\|}, \frac{\phi(\boldsymbol{u}')}{\|\phi(\boldsymbol{u}')\|} \right\rangle = \frac{\kappa(\boldsymbol{u}, \boldsymbol{u}')}{\sqrt{\kappa(\boldsymbol{u}, \boldsymbol{u})\kappa(\boldsymbol{u}', \boldsymbol{u}')}}. \tag{2.28}$$

The concept of kernel function gives rise to the Gram matrix defined as follows:

Definition 2.3.2 (Gram matrix) *Given a kernel function $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ and input data $\{\boldsymbol{u}_n\}_{n=1}^{M}$, the $(M \times M)$ matrix $\boldsymbol{K}$ with arbitrary entry*

$$\boldsymbol{K}_{ij} = \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) \quad \text{for } i, j = 1, ..., M \tag{2.29}$$

is called the Gram matrix or kernel matrix with respect to $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_M$.

The Gram matrix, which contains information on inner products between training data is significant in all kernel-based algorithms. It can also be viewed as an interface between the input data and the learning algorithms. Additionally in matrix analysis literature, positive definite matrix is defined as:

Definition 2.3.3 (Positive definite matrix) *A real symmetric* $(M \times M)$ *matrix* $\boldsymbol{K}$ *satisfying*

$$\sum_{i=1}^{M}\sum_{j=1}^{M} c_i c_j \, \boldsymbol{K}_{ij} \geq 0 \tag{2.30}$$

for all $c_i \in \mathbb{R}$ *is said to be positive definite.*

As the term positive definite matrix already defined, we will straightforwardly introduce the positive definite property in the context of kernel theory.

Definition 2.3.4 (Positive-definite kernel) *A kernel function* $\kappa(\boldsymbol{u}, \boldsymbol{u}')$ *on* $\mathcal{U} \times \mathcal{U}$ *is called a positive-definite kernel if*

1. *it is symmetric:* $\kappa(\boldsymbol{u}, \boldsymbol{u}') = \kappa(\boldsymbol{u}', \boldsymbol{u})$*;*
2. *the Gram matrix is positive definite:*

$$\boldsymbol{\alpha}^\top \boldsymbol{K} \boldsymbol{\alpha} = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \, \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) \geq 0 \tag{2.31}$$

with any vector $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_n]^\top \in \mathbb{R}^n$ *and set* $\{\boldsymbol{u}_i\}_{i=1,\ldots,n}$*. It is strictly positive-definite only if, for* $\boldsymbol{\alpha} = 0$

$$\boldsymbol{\alpha}^\top \boldsymbol{K} \boldsymbol{\alpha} = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \, \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = 0. \tag{2.32}$$

Positive-definite (PD) kernels can be regarded as generalized dot products. Indeed, any dot product is a PD kernel. However, linearity in the arguments, which is a standard property of dot products, does not carry over to general kernels. Another property of dot products, the Cauchy-Schwarz inequality, does have a natural generalization: if κ is a PD kernel, and $\boldsymbol{u}_1, \boldsymbol{u}_2 \in \mathcal{U}$, then

$$|\kappa(\boldsymbol{u}_1, \boldsymbol{u}_2)|^2 \leq \kappa(\boldsymbol{u}_1, \boldsymbol{u}_1) \, \kappa(\boldsymbol{u}_2, \boldsymbol{u}_2). \tag{2.33}$$

It can be shown that if κ is a complex PD kernel, its real part is a real-valued PD kernel. From now on, we only focus on positive-definite kernels, and simply refer to them as kernels. Definition 2.3.4 ensures that the Gram matrix is symmetric positive definite.

Definition 2.3.5 (Reproducing kernel or Mercer kernel [Mercer 1909]) *A reproducing kernel over the set* $\mathcal{U}$ *is a continuous, symmetric, positive-definite function* $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$*, possessing the so-called producing property:*

$$f(\boldsymbol{u}) = \langle f, \kappa(\cdot, \boldsymbol{u}) \rangle_{\mathcal{H}}, \quad \text{for all } f \in \mathcal{H}, \boldsymbol{u} \in \mathcal{U}, \tag{2.34}$$

in particular $\kappa(\boldsymbol{u}, \boldsymbol{u}') = \langle \kappa(\cdot, \boldsymbol{u}), \kappa(\cdot, \boldsymbol{u}') \rangle_{\mathcal{H}}$*.*

It is well known now that the above two kernel definitions, either as a positive-definite kernel, or as a reproducing kernel, are equivalent. Furthermore, it was proved that there is a one to one correspondence between the space of positive-definite kernels and the space of reproducing kernel Hilbert spaces.

2.3.2 Reproducing kernel Hilbert spaces (RKHS)

A vector space $\mathcal{U}$ over the real field $\mathbb{R}$ is an inner product space if there exists a real-valued symmetric bilinear map, i.e., inner or dot product $\langle \cdot, \cdot \rangle$, that satisfies $\langle \boldsymbol{u}, \boldsymbol{u} \rangle \geq 0$. Furthermore, we will say the inner product is strict if $\langle \boldsymbol{u}, \boldsymbol{u} \rangle = 0$ if and only if $\boldsymbol{u} = 0$. Although an inner product space is sometimes referred to as a Hilbert space, its formal definition still requires additional properties of completeness and separability, as well as the infinite dimension.

Definition 2.3.6 (Hilbert space [Shawe-Taylor 2004]) *A Hilbert space $\mathcal{H}$ is an inner product space with the additional properties that it is separable and complete. Completeness refers to the property that every Cauchy sequence $\{h_n\}_{n \geq 1}$ of elements of $\mathcal{H}$ converges to an element $h \in \mathcal{H}$, where a Cauchy sequence is a sequence satisfying the property that*

$$\sup_{m>n} \|h_n - h_m\| \to 0, \quad \text{as } n \to \infty. \tag{2.35}$$

A space $\mathcal{H}$ is separable if for any $\epsilon > 0$ there is a finite set of elements $h_1, \ldots, h_N$ of $\mathcal{H}$ such that for all $h \in \mathcal{H}$

$$\min_n \|h - h_n\| < \epsilon. \tag{2.36}$$

The reason for the importance of completeness and separability properties is that they ensure that the feature space is a complete, separable inner product space.

Definition 2.3.7 (Reproducing kernel Hilbert space [Aronszajn 1950]) *Consider a linear space $\mathcal{H}$ of real-valued functions, f defined on a set $\mathcal{U}$. Suppose that in $\mathcal{H}$ we can define an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ with corresponding norm $\|\cdot\|_{\mathcal{H}}$ and that $\mathcal{H}$ is complete with respect to that norm, i.e., $\mathcal{H}$ is a Hilbert space. We call $\mathcal{H}$ a reproducing kernel Hilbert space, if there exists a function κ: $\mathcal{U} \times \mathcal{U} \to \mathbb{R}$ with the following properties:*

1. *For every $\boldsymbol{u} \in \mathcal{U}$, $\kappa(\cdot, \boldsymbol{u})$ belongs to $\mathcal{H}$ (or equivalently κ spans $\mathcal{H}$, i.e. $\mathcal{H} = \overline{span\{\kappa(\cdot, \boldsymbol{u}),\ \boldsymbol{u} \in \mathcal{U}\}}$, where the overline denotes the closure of a set).*
2. *κ has the reproducing property.*

Moreover, κ is a positive-definite kernel and the mapping ψ: $\mathcal{U} \to \mathcal{H}$, with $\psi(\boldsymbol{u}) = \kappa(\cdot, \boldsymbol{u})$, for all $\boldsymbol{u} \in \mathcal{U}$ is called the feature map of $\mathcal{H}$. It can be found that the RKHS uniquely determines κ. The RKHS sometimes roughly equivalent to a space of functions with an inner product, has already been shown to play an important role in kernel-based methods. Linear processing successfully performed in RKHS $\mathcal{H}$ by mapping the data into a higher dimensional or possibly infinite feature space, has been proven to be a very powerful tool to address nonlinear problems in original space.

Theorem 2.3.8 (Riesz representation [Riesz 1907]) *Let $\mathcal{H}$ be a general Hilbert space and let $\mathcal{H}^*$ denote its dual space. Every elements Φ of $\mathcal{H}^*$ can be uniquely expressed in the form:*

$$\Phi(f) = \langle f, \phi \rangle_{\mathcal{H}}, \tag{2.37}$$

for some $\phi \in \mathcal{H}$. Moreover, $\|\Phi\|_{\mathcal{H}^*} = \|\phi\|_{\mathcal{H}}$.

According to the Riesz representation, we have that for every $\boldsymbol{u} \in \mathcal{U}$, there exists an unique element $\kappa(\cdot, \boldsymbol{u}) \in \mathcal{H}$, such that for every $f \in \mathcal{H}$, $f(\boldsymbol{u}) = \langle f, \kappa(\cdot, \boldsymbol{u})\rangle_{\mathcal{H}}$. As a consequence, replacing f by $\kappa(\cdot, \boldsymbol{u}')$ we immediately have $\kappa(\boldsymbol{u}', \boldsymbol{u}) = \langle \kappa(\cdot, \boldsymbol{u}'), \kappa(\cdot, \boldsymbol{u})\rangle$.

The following popular theorem establishes that a RKHS may have infinite dimension, however, the solution of any regularized regression optimization problem lies in the span of n particular kernels.

Theorem 2.3.9 (Representer theorem [Schölkopf 2000]) *Suppose we are given a nonempty set $\mathcal{U}$, a positive-definite real-valued kernel κ on $\mathcal{U} \times \mathcal{U}$, a training set $\{\boldsymbol{u}_i, d_i\}_{i=1,\ldots,n} \in \mathcal{U} \times \mathbb{R}$, a strictly monotonically increasing real-valued function g on $[0, \infty]$, an arbitrary cost function L: $(\mathcal{U} \times \mathbb{R}^2)^n \to \mathbb{R} \cup \{\infty\}$, and a class of functions*

$$\mathcal{F} = \left\{ f \in \mathbb{R}^{\mathcal{U}} \middle| f(\cdot) = \sum_{i=1}^{\infty} \alpha_i \, \kappa(\cdot, \boldsymbol{x}_i), \alpha_i \in \mathbb{R}, \boldsymbol{x}_i \in \mathcal{U}, \|f\| < \infty \right\} \tag{2.38}$$

where $\|\cdot\|$ is the norm in the RKHS $\mathcal{H}$ associated with κ,

$$\left\| \sum_{i=1}^{\infty} \alpha_i \kappa(\cdot, \boldsymbol{x}_i) \right\|^2 = \sum_{i,j=1}^{\infty} \alpha_i \alpha_j \, \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{2.39}$$

Then any $f \in \mathcal{F}$ minimizing the regularized risk functional

$$L\big((\boldsymbol{u}_1, d_1, f(\boldsymbol{u}_1)), \ldots, (\boldsymbol{u}_n, d_n, f(\boldsymbol{u}_n))\big) + g(\|f\|) \tag{2.40}$$

admits a representation of the form $f(\cdot) = \sum_{i=1}^{n} \alpha_i \, \kappa(\cdot, \boldsymbol{u}_i)$.

The significance of the Representer theorem is that it demonstrates that a whole range of learning algorithms have optimal solutions that can be expressed as an expansion in terms of kernel evaluation of the training examples.

Theorem 2.3.10 (Moore [Moore 1916]) *For any reproducing kernel κ over the set $\mathcal{U}$, there exists an unique Hilbert space of function on $\mathcal{H}$ for which κ is a reproducing kernel.*

Notice that Moore theorem establishes a one-to-one correspondence between RKHS on a set and positive definite function on the set.

Proposition 2.3.11 (Kernel trick [Scholkopf 2001]) *Given an algorithm which is formulated in terms of a positive kernel $\tilde{\kappa}$, one can construct an alternative algorithm by replacing $\tilde{\kappa}$ by another kernel κ.*

The best known application of the kernel trick is in the case where $\tilde{\kappa}$ and κ are the common dot product in the input domain and the selected kernel function, respectively. Since the structure of the algorithm remains exactly the same, the nonlinearity is then obtained at no computational cost. The operation that transforms linear algorithms into their nonlinear counterparts using kernel is often called kernelization. The kernel trick is not limited to the classical case. Hence, $\tilde{\kappa}$ and κ can both be nonlinear kernels corresponding to different RKHS, namely, different nonlinearities.

Kernel-based methods are a powerful nonparametric modeling tool, whose essence is transforming the data in lower dimensional input space into a high or infinite dimensional feature space via a reproducing kernel such that the inner product operation can be computed efficiently through the kernel evaluations. In the machine learning literature, it has been proven that most linear learning algorithms can be applied to naturally solve nonlinear problems within high-dimensional RKHS using the kernel trick. In Figure 2.3, a two-dimensional nonlinear regression problem boils down to the three-dimensional linear regression problem preprocessed by a nonlinear map $\psi : [u_1, u_2]^\top \to [u_1^2, u_2^2, \sqrt{2}u_1u_2]^\top$.

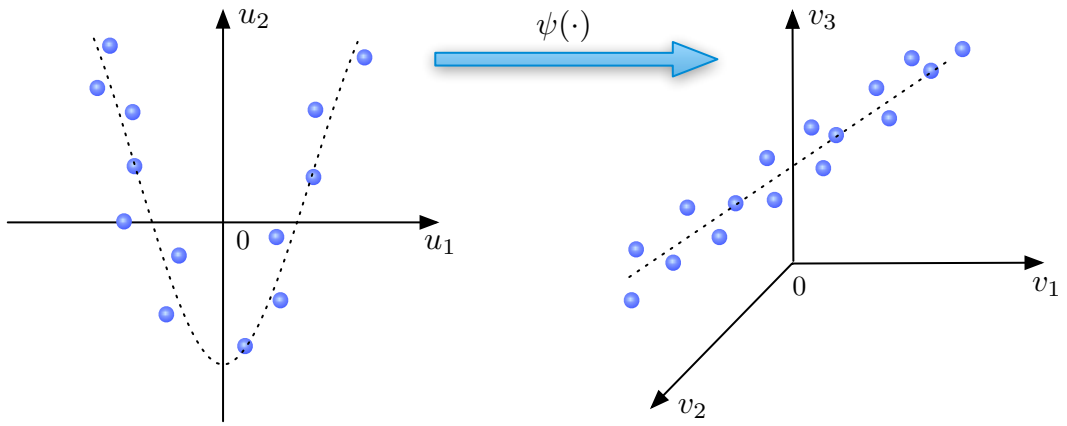


Figure 2.3: Example of nonlinear regression problem solved in a feature space with the linear method.

2.3.3 Examples of kernel

The kernel function replacing the inner product of two feature vectors commonly corresponding to two inputs, is an attractive computational shortcut to create complicated feature spaces. In practical approaches directly choosing or defining a kernel function is equivalent to implicitly defining a feature space in which the constructed algorithms are performed. Next we shall introduce two widely-used types of basic positive kernel functions.

2.3.3.1 Radial basis function kernels

Radial basis function (RBF) kernels are kernels that can be written in the form

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = f\left(d(\boldsymbol{u}, \boldsymbol{u}')\right) \tag{2.41}$$

where d is a metric on $\mathcal{U}$, and $f(\cdot)$ is a function on $\mathbb{R}_+$. The metric is usually defined as the dot product, that is, $d = \|\boldsymbol{u} - \boldsymbol{u}'\| = \sqrt{\langle \boldsymbol{u} - \boldsymbol{u}', \boldsymbol{u} - \boldsymbol{u}' \rangle}$. In this case, RBF kernels are translation invariant, $\kappa(\boldsymbol{u}, \boldsymbol{u}') = \kappa(\boldsymbol{u} + \boldsymbol{u}_0, \boldsymbol{u}' + \boldsymbol{u}_0)$ for all $\boldsymbol{u}_0 \in \mathcal{U}$. In addition, the RBF kernels have the property of unitary invariance, $\kappa(\boldsymbol{u}, \boldsymbol{u}') = \kappa(\boldsymbol{T}\boldsymbol{u}, \boldsymbol{T}\boldsymbol{u}')$ if $\boldsymbol{T}^\top = \boldsymbol{T}^{-1}$ is a rotation matrix. Some typical examples of RBF kernel functions include:

- Gaussian kernel is defined as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \exp\left(-\frac{\|\boldsymbol{u} - \boldsymbol{u}'\|^2}{2\xi^2}\right) \tag{2.42}$$

  which is sometimes presented as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \exp\left(-\zeta\|\boldsymbol{u} - \boldsymbol{u}'\|^2\right) \tag{2.43}$$

  where ξ is the kernel bandwidth that specifies the precise shape of the kernel function. By (2.42) and (2.43), the parameter ζ is clearly equal to $1/2\xi^2$.

- Locally Gaussian kernel is defined as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \max\left(0, 1 - \frac{\|\boldsymbol{u} - \boldsymbol{u}'\|}{6\xi^2}\right)^r \exp\left(-\frac{\|\boldsymbol{u} - \boldsymbol{u}'\|^2}{2\xi^2}\right) \tag{2.44}$$

  with parameter $r > 0$.

  Theorem 2.3.12 *Suppose that $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_n \in \mathcal{U}$ are distinct input data. The Gram matrix $\boldsymbol{R}_{\kappa\kappa}$ built with the Gaussian kernel has full rank [Micchelli 1986].*

  In other words, the transformed points $\psi(\boldsymbol{u}_1), \psi(\boldsymbol{u}_2), \ldots, \psi(\boldsymbol{u}_n)$ are linearly independent and span an n-dimensional subspace of $\mathcal{H}$.

- Exponential kernel is defined as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \exp\left(-\frac{\|\boldsymbol{u} - \boldsymbol{u}'\|}{2\xi^2}\right). \tag{2.45}$$

  Alternatively, it could also be written as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \exp\left(-\zeta\|\boldsymbol{u} - \boldsymbol{u}'\|\right). \tag{2.46}$$

  Observe that the exponential kernel is closely related to the Gaussian kernel function except of the square of the ℓ_2-norm.

- The Laplacian Kernel is defined as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \exp\left(\frac{\|\boldsymbol{u} - \boldsymbol{u}'\|}{\xi}\right) \tag{2.47}$$

  which is completely equivalent to the exponential kernel, neglecting of being less sensitive to the changes in the parameter ξ.

Because the adjustable kernel bandwidth plays a major role in the performance of kernel-based methods, it needs to be carefully tuned to the particular problem. Hence, the selection of kernel bandwidth is a key point of research for kernel adaptive filters with the RBF kernel.

2.3.3.2 Projective basis function kernels

Projective basis function (PBF) kernels are of the form:

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = g\big(\alpha\, \boldsymbol{u}^\top \boldsymbol{u}' + c\big) \tag{2.48}$$

where $g(\cdot)$ is a general transfer function, and optional constant $c \geq 0$. Examples of PBF kernels determined by different classes of transfer function $g(\cdot)$ are listed as follows:

- The inhomogeneous polynomial kernel is a non-stationary kernel, which is defined as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \big(\alpha\, \boldsymbol{u}^\top \boldsymbol{u}' + c\big)^p \tag{2.49}$$

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

- When $p = 1$ we obtain the simplest linear kernel, which is expressed as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \alpha\, \boldsymbol{u}^\top \boldsymbol{u}' + c \tag{2.50}$$

  where α is an adjustable slope. Kernel learning algorithms using a linear kernel are degenerated into their non-kernel or linear counterparts, which is beneficial to the consideration of the mixture of linear and nonlinear scenarios.

- If neglecting the constant c, the inhomogeneous polynomial kernel reduces to the homogenous polynomial kernel, which means that

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \big(\alpha\, \boldsymbol{u}^\top \boldsymbol{u}'\big)^p. \tag{2.51}$$

- Cosine kernel is defined as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \frac{\boldsymbol{u}^\top \boldsymbol{u}'}{\|\boldsymbol{u}\|\|\boldsymbol{u}'\|}. \tag{2.52}$$

- Correlation kernel is defined as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \exp\left(\frac{\boldsymbol{u}^\top \boldsymbol{u}'}{\|\boldsymbol{u}\|\|\boldsymbol{u}'\|} - c\right). \tag{2.53}$$

- Hyperbolic tangent kernel is also known as the sigmoid kernel and as the multilayer perceptron (MLP) kernel, which is defined as

$$\kappa(\boldsymbol{u}, \boldsymbol{u}') = \tanh\Big(\alpha\, \boldsymbol{u}^\top \boldsymbol{u}' + c\Big). \tag{2.54}$$

  The sigmoid kernel function comes from the neural networks field, where the bipolar sigmoid function is often used as an activation function for artificial neurons. Thus this kernel is widely used in support vector machines due to its origin from neural network theory.

2.3.4 Kernel construction

In this last subsection, we describe some well-known basic kernel functions. Since designing a suitable kernel function for an input space is much more straightforward than manipulating the complex feature space, we will discuss the ways of how to construct more complicated and useful kernels by combining simpler ones.

Let κ_1 and κ_2 be symmetric and positive-definite kernels over $\mathcal{U} \times \mathcal{U}$, $\mathcal{U} \subseteq \mathbb{R}^n$, $f(\cdot)$ a real-valued function on $\mathcal{U}$, $\phi : \mathcal{U} \to \mathbb{R}^N$ with κ_3 a kernel over $\mathbb{R}^N \times \mathbb{R}^N$, and a $(n \times n)$ dimensional symmetric positive semi-definite matrix $\boldsymbol{G}$. Then the following functions are valid kernels:

1. $\kappa(\boldsymbol{u}, \boldsymbol{u}') = c_1\, \kappa_1(\boldsymbol{u}, \boldsymbol{u}') + c_2\, \kappa_2(\boldsymbol{u}, \boldsymbol{u}')$ for $c_1, c_2 \geq 0$.
   Proof. Suppose that we have $c_1\, \kappa_1(\boldsymbol{u}, \boldsymbol{u}') = \langle \sqrt{c_1}\phi_1(\boldsymbol{u}), \sqrt{c_1}\phi_1(\boldsymbol{u}') \rangle$ and $c_2\, \kappa_2(\boldsymbol{u}, \boldsymbol{u}') = \langle \sqrt{c_2}\phi_2(\boldsymbol{u}), \sqrt{c_2}\phi_2(\boldsymbol{u}') \rangle$, then:

$$\begin{aligned} \kappa(\boldsymbol{u}, \boldsymbol{u}') &= c_1\, \kappa_1(\boldsymbol{u}, \boldsymbol{u}') + c_2\, \kappa_2(\boldsymbol{u}, \boldsymbol{u}') \\ &= \big\langle \sqrt{c_1}\phi_1(\boldsymbol{u}), \sqrt{c_1}\phi_1(\boldsymbol{u}') \big\rangle + \big\langle \sqrt{c_2}\phi_2(\boldsymbol{u}), \sqrt{c_2}\phi_2(\boldsymbol{u}') \big\rangle \\ &= \big\langle [\sqrt{c_1}\phi_1(\boldsymbol{u}) \sqrt{c_2}\phi_2(\boldsymbol{u}')], [\sqrt{c_1}\phi_1(\boldsymbol{u}) \sqrt{c_2}\phi_2(\boldsymbol{u}')] \big\rangle. \end{aligned} \tag{2.55}$$

   We can see that the function $\kappa(\boldsymbol{u}, \boldsymbol{u}')$ can be expressed as an inner product. □
   The construction of kernel functions by summing several basic kernels is a simple and effective modeling method within a wide variety of contexts. Kernels κ_1 and κ_2 can be of different types, allowing us to model the input data as a sum of independent functions representing the different types of structure such as linear kernel, periodic kernel, Gaussian kernel, etc.

2. $\kappa(\boldsymbol{u}, \boldsymbol{u}') = \kappa_1(\boldsymbol{u}, \boldsymbol{u}')\, \kappa_2(\boldsymbol{u}, \boldsymbol{u}')$.
   Proof. Note that the Gram matrix $\boldsymbol{K}$ for κ is the Hadamard product of $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$, namely, $\boldsymbol{K} = \boldsymbol{K}_1 \odot \boldsymbol{K}_2$. Suppose that $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ are covariance matrices of $[x_1, \ldots, x_n]^\top$ and $[y_1, \ldots, y_n]^\top$, respectively. Then matrix $\boldsymbol{K}$ is simply the covariance matrix of $[x_1 y_1, \ldots, x_n y_n]^\top$, implying that it is symmetric and positive-definite. □
   Multiplying two positive-definite kernels always results in another positive-definite kernel function. Furthermore, the properties of constructed kernel are determined by the used kernels as in the additivity case.

3. $\kappa(\boldsymbol{u}, \boldsymbol{u}') = f(\boldsymbol{u}) f(\boldsymbol{u}')$, for a real-valued function f: $\mathcal{U} \to \mathbb{R}$.
   Proof. We can rearrange the bilinear form as follows:

$$\begin{aligned} \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j\, \boldsymbol{K} &= \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j\, f(\boldsymbol{u}) f(\boldsymbol{u}') \\ &= \sum_{i=1}^{n} c_i\, f(\boldsymbol{u}) \sum_{j=1}^{n} c_j\, f(\boldsymbol{u}') \\ &= \Big\| \sum_{i=1}^{n} c_i\, f(\boldsymbol{u}) \Big\|^2 \geq 0 \end{aligned} \tag{2.56}$$

are required. □

4. $\kappa(\boldsymbol{u}, \boldsymbol{u}') = \kappa_3\big(\phi(\boldsymbol{u}), \phi(\boldsymbol{u}')\big)$, where ϕ: $\mathcal{U} \to \mathbb{R}^N$.
PROOF. Since κ_3 is a kernel function, the matrix obtained by restricting κ_3 to the points $\phi(\boldsymbol{u}_1), \ldots, \phi(\boldsymbol{u}_n)$ is positive semi-definite as required. □

5. $\kappa(\boldsymbol{u}, \boldsymbol{u}') = g\big(\kappa_1(\boldsymbol{u}, \boldsymbol{u}')\big)$, where $g(\cdot)$ is a polynomial with positive coefficients.
PROOF. Since each polynomial term is a product of kernels with a positive coefficient, the proof follows by applying 1 and 2. □

In order to capture the structure of input data, one can manipulate different types and any number of individual single kernels together using the above mentioned rules to create the required kernel function.

2.4 The existing kernel adaptive filtering algorithms

Functional characterization of an unknown system usually begins by observing the response of that system to input signals. Information obtained from such observations can then be used to derive a model. As illustrated by the block diagram in Figure 2.4, the goal of system identification is to use pairs $(\boldsymbol{u}_n, d_n)$ of inputs and noisy outputs to derive a function that maps an arbitrary system input $\boldsymbol{u}_n$ into an appropriate output $\hat{d}_n$. Dynamic system identification has played a crucial role in the development of techniques for stationary and non-stationary signal processing. Adaptive algorithms use an error signal e_n to adjust the model coefficients $\boldsymbol{\alpha}_n$, in an online way, in order to minimize a given objective function. Most existing approaches focus on linear models due to their inherent simplicity from points of view of concept and implementation. However, there are many practical situations, e.g., in communications and biomedical engineering, where the nonlinear processing of signals is needed. Unlike linear systems which can be uniquely identified by their impulse response, nonlinear systems can be characterized by representations ranging from higher-order statistics, e.g., [Nam 1994, Nikias 1993], to series expansion methods, e.g., [Schetzen 1980, Wiener 1958]. Polynomial filters, usually called Volterra series based filters [Mathews 2000], and neural networks [Haykin 1999] have been extensively studied over the years. Volterra filters are attractive because their output can be expressed as a linear combination of nonlinear functions of the input signal, which simplifies the design of gradient-based and recursive least squares adaptive algorithms. Nevertheless, the considerable number of parameters to estimate, which goes up exponentially as the order of the nonlinearity increases is a severe drawback. Neural networks are proven to be universal approximators under suitable conditions [Kolmogorov 1957]. It is, however, well known that algorithms used for neural network training suffer from problems such as being trapped into local minima, slow convergence and great computational requirements.

Recently, the adaptive filtering in RKHS has become an appealing tool in many fields. This framework for nonlinear system identification consists of mapping the original input

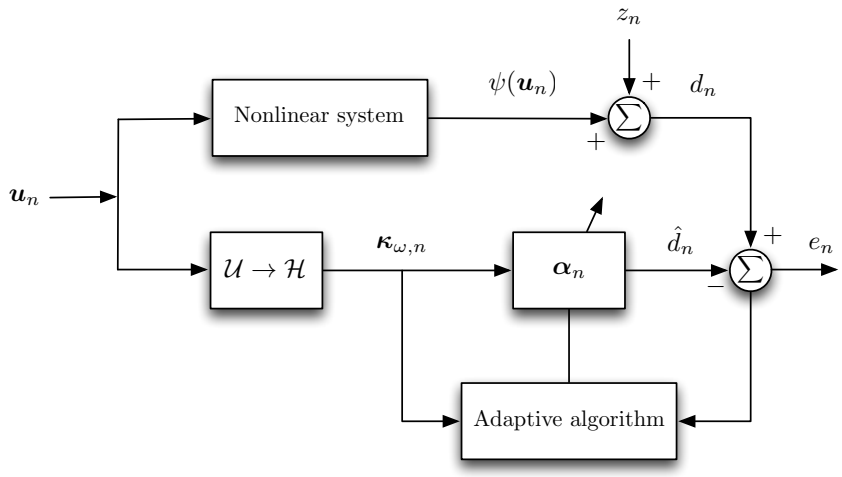


Figure 2.4: Kernel-based adaptive system identification.

data $\boldsymbol{u}_n$ into a RKHS $\mathcal{H}$, and applying a linear adaptive filtering technique to the resulting functional data. The block diagram presented in Figure 2.4 presents the basic principles of this strategy. The input space $\mathcal{U}$ is a compact of $\mathbb{R}^q$, $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ is a reproducing kernel as listed in previous section, and $(\mathcal{H}, \langle \cdot, \cdot \rangle_{\mathcal{H}})$ is the induced RKHS with its inner product. The additive noise z_n is assumed to be white and zero-mean, with variance σ_z^2. Considering the least-squares approach, given N input vectors $\boldsymbol{u}_n$ and desired outputs d_n, the identification problem consists of determining the optimum function $\psi_{\text{opt}}(\cdot)$ in $\mathcal{H}$ that solves the problem

$$\psi_{\text{opt}} = \underset{\psi \in \mathcal{H}}{\arg\min} \left\{ J(\psi) = \sum_{i=1}^{N} (d_i - \psi(\boldsymbol{u}_i))^2 + \gamma\, \Omega(\|\psi\|) \right\} \tag{2.57}$$

with $\Omega(\cdot)$ a real-valued monotonic regularizer on $\mathbb{R}_+$ and γ a positive regularization constant. By virtue of the Representer theorem [Schölkopf 2000], the function $\psi(\cdot)$ can be written as a kernel expansion in terms of available training data, namely, $\psi(\cdot) = \sum_{j=1}^{N} \alpha_j\, \kappa(\cdot, \boldsymbol{u}_j)$. Therefore, the above optimization problem becomes

$$\boldsymbol{\alpha}_{\text{opt}} = \underset{\boldsymbol{\alpha} \in \mathbb{R}^N}{\arg\min} \left\{ J(\boldsymbol{\alpha}) = \sum_{j=1}^{N} (d_j - \boldsymbol{\alpha}^\top \boldsymbol{\kappa}_j)^2 + \gamma\, \Omega(\boldsymbol{\alpha}) \right\} \tag{2.58}$$

where $\boldsymbol{\kappa}_j$ is the $(N \times 1)$ vector with the i-th entry $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$. Online processing of time series data raises the question of how to process an increasing amount N of observations as new data is collected. Indeed, an undesirable characteristic of problem (2.57)–(2.58) is that the order of the filters grows linearly with the number of input data. This dramatically increases the computational burden and memory requirement of nonlinear system identification methods. To overcome this drawback, several authors have focused on the finite-order models of the form

$$\psi(\cdot) = \sum_{m=1}^{M} \alpha_m\, \kappa(\cdot, \boldsymbol{u}_{\omega_m}). \tag{2.59}$$

Definition 2.4.1 (Dictionary) *The set of support vectors denoted by* $\mathcal{D} = \{\kappa(\cdot, \boldsymbol{u}_{\omega_m})\}_{m=1}^{M}$ *is called the dictionary, which has to be learnt from input data.*

The dimension of the dictionary M is analogous to the order of linear transversal filters. The subscript ω_m allows us to clearly distinguish dictionary elements (or support vectors) $\boldsymbol{u}_{\omega_m}, \ldots, \boldsymbol{u}_{\omega_M}$ from ordinary input data $\boldsymbol{u}_n$. Online identification of kernel-based models generally relies on a two-step process at each iteration: a model order control step that updates the dictionary, and a weight parameters update step. This two-step process is the essence of most adaptive filtering techniques with kernels [Liu 2010].

2.4.1 Sparsification criteria

In general the observed data are not equally informative, for instance, in SVM only the data that are close to the boundary are important relative to the others. Moreover, the same data may convey a different amount of information depending on the sequence of handling by the learning system. Intuitively, the subsequent data become redundant to learning system after processing sufficient samples from the same source. Hence a subset of data must be selected for efficient training and sparse representation especially in a sequential learning setting. In addition, the most important point is that online finite-order model (2.59) needs to be controlled by some sparsification rule. In the following, we shall recall several well-known sparsification criteria of dictionary commonly used in kernel adaptive filtering.

2.4.1.1 Novelty criterion (NC)

The novelty criterion is a simple way to examine whether the newly arrived data is informative enough to be added into the online dictionary [Platt 1991]. The two types of measures for latest data candidate must simultaneously meet the conditions: i) the distance to all the elements in dictionary is larger than the given threshold δ_ω; ii) the prediction error is greater than another preset threshold δ_e; summarized as follows:

$$\begin{aligned} \min_{\boldsymbol{u}_{\omega_m} \in \mathcal{D}} \|\boldsymbol{u}_{\omega_m} - \boldsymbol{u}_n\| &\geq \delta_\omega; \\ \|d_n - \hat{d}_n\| &\geq \delta_e. \end{aligned} \tag{2.60}$$

Otherwise, the new data $\boldsymbol{u}_n$ should be rejected.

2.4.1.2 Approximate linear dependency (ALD)

The ALD sparsification criterion was proposed in [Engel 2004]. When the input data pair is available, the ALD criterion may be expressed as

$$\min_{\forall \boldsymbol{w} \in \mathbb{R}^M} \left\| \boldsymbol{h}_n^\top \boldsymbol{w} - \kappa(\cdot, \boldsymbol{u}_n) \right\| \leq \delta_a \tag{2.61}$$

with kernel evaluation vector $\boldsymbol{h}_n = [\kappa(\cdot, \boldsymbol{u}_{\omega_1}), \ldots, \kappa(\cdot, \boldsymbol{u}_{\omega_M})]^\top$, the coefficients vector $\boldsymbol{w}$ and the threshold δ_a. The arbitrary entry of Gram matrix $\boldsymbol{K}$ using the elements in dictionary

is written as $\boldsymbol{K}_{ij} = \kappa(\boldsymbol{u}_{\omega_i}, \boldsymbol{u}_{\omega_j})$ with $1 \leq i, j \leq M$. Assuming that the Gram matrix $\boldsymbol{K}$ is inevitable, the optimal coefficient vector can be calculated as

$$\boldsymbol{w}_{\text{opt}} = \boldsymbol{K}^{-1}\boldsymbol{h}_n. \tag{2.62}$$

By replacing $\boldsymbol{w}$ in (2.61) with $\boldsymbol{w}_{\text{opt}}$, the ALD criterion can be reformulated as

$$\boldsymbol{h}_n\boldsymbol{K}^{-1}\boldsymbol{h}_n - \kappa(\boldsymbol{u}_n, \boldsymbol{u}_n) \leq \delta_a. \tag{2.63}$$

The above condition indicates that if the input data is approximately linear dependent with respect to the current dictionary, it turns out to be redundant and will be not accepted. The computational complexity of the ALD criterion is quadratic $O(M^2)$ at each time instant n using matrix inversion lemma, relative to the length of online dictionary M.

2.4.1.3 Coherence sparsification (CS)

Coherence is a fundamental parameter to characterize a dictionary in linear sparse approximation problems. Within the context of the KAF, it is defined as [Richard 2009]

$$\delta = \max_{i \neq j} |\kappa(\boldsymbol{u}_{\omega_i}, \boldsymbol{u}_{\omega_j})| \tag{2.64}$$

where κ is an unit-norm kernel. Coherence criterion suggests inserting the candidate input $\kappa(\cdot, \boldsymbol{u}_n)$ into the dictionary provided that its coherence remains below a given threshold δ_0

$$\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \delta_0, \tag{2.65}$$

where δ_0 is a parameter in $[0, 1[$ determining both the level of sparsity and the coherence of the dictionary. Moreover, it was proven that the dimension of the dictionary generated by the CS criterion remains finite or upper bounded as n goes infinite for any input sequence [Richard 2009].

All the above-mentioned criteria use different learning strategies to decide, at each time instant n, whether $\kappa(\cdot, \boldsymbol{u}_n)$ deserves to be inserted into the dictionary or not. Other well-known criteria include the surprise criterion [Liu 2009a], closed-ball sparsification criterion [Slavakis 2013], presence-based sparsification criterion [Tobar 2014a], etc., but they are not detailed now for brevity. Without loss of generality, we focus on the CS criterion for the kernel adaptive filtering algorithms presented in this dissertation due to its simplicity and effectiveness. However, any other dictionary update criterion could be considered among of the above-mentioned ones without too much effort.

2.4.2 Kernel affine projection algorithm

There is no unique and omnipotent solution to all kinds of nonlinear adaptive filtering problems as in linear situations. We also have a “kit of tools" including of a variety of kernel adaptive filtering algorithms inspired by classical linear adaptive filters, each of which accordingly offers superior features and inevitable defects of its own.

Affine projection algorithm determines a projection of the solution vector $\boldsymbol{\alpha}$ that solves an under-determined least-squares problem. At each time step n, only the p most recent

inputs $\{\boldsymbol{u}_n, \ldots, \boldsymbol{u}_{n-p+1}\}$ and observations $\{d_n, \ldots, d_{n-p+1}\}$ are used. Let $\boldsymbol{H}_n$ be the matrix whose (i,j)-th entry is $\kappa(\boldsymbol{u}_{n-i+1}, \boldsymbol{u}_{\omega_j})$, and $\boldsymbol{d}_n$ is the column vector whose i-th element is d_{n-i+1}. The kernel affine projection problem at time instant n can be formulated as

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}_{n-1}\|^2 \qquad \text{subject to} \qquad \boldsymbol{d}_n = \boldsymbol{H}_n\boldsymbol{\alpha}. \tag{2.66}$$

In other words, $\boldsymbol{\alpha}_n$ is obtained by projecting $\boldsymbol{\alpha}_{n-1}$ onto the intersection of the p manifolds $\mathcal{A}_i$ defined as

$$\mathcal{A}_i = \{\boldsymbol{\alpha} : \boldsymbol{\kappa}_{n-i+1}\boldsymbol{\alpha} - d_{n-i+1} = 0\}, \qquad \text{for } i = 1, \ldots, p \tag{2.67}$$

with $\boldsymbol{\kappa}_{n-i+1} = [\kappa(\boldsymbol{u}_{n-i+1}, \boldsymbol{u}_{\omega_1}), \ldots, \kappa(\boldsymbol{u}_{n-i+1}, \boldsymbol{u}_{\omega_M})]^\top$. When the new data $\boldsymbol{u}_n$ is available, there are two possible situations according to the coherence sparsification criterion in the following.

- **Rejection case:** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| > \delta_0$
  In this case $\kappa(\cdot, \boldsymbol{u}_n)$ can be reasonably well represented by the kernel functions already in the dictionary. Thus, it does not need to be inserted into the dictionary. The solution to (2.66) can be obtained by minimizing the Lagrangian function

  $$J(\boldsymbol{\alpha}, \lambda) = \|\boldsymbol{\alpha} - \boldsymbol{\alpha}_{n-1}\|^2 + \boldsymbol{\lambda}^\top(\boldsymbol{d}_n - \boldsymbol{H}_n\boldsymbol{\alpha}) \tag{2.68}$$

  where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers. Differencing this equation with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$, and setting the derivatives to zero, we get

  $$2(\boldsymbol{\alpha}_n - \boldsymbol{\alpha}_{n-1}) = \boldsymbol{H}_n^\top\boldsymbol{\lambda} \tag{2.69}$$

  $$\boldsymbol{H}_n\boldsymbol{\alpha}_n = \boldsymbol{d}_n. \tag{2.70}$$

  Assuming $\boldsymbol{H}_n\boldsymbol{H}_n^\top$ to be nonsingular, these equations lead to $\boldsymbol{\lambda} = 2(\boldsymbol{H}_n\boldsymbol{H}_n^\top)^{-1}(\boldsymbol{d}_n - \boldsymbol{H}_n\boldsymbol{\alpha}_{n-1})$. Substituting this into (2.69), we obtain a recursive update equation for $\boldsymbol{\alpha}$

  $$\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n-1} + \eta\,\boldsymbol{H}_n^\top(\epsilon\boldsymbol{I} + \boldsymbol{H}_n\boldsymbol{H}_n^\top)^{-1}(\boldsymbol{d}_n - \boldsymbol{H}_n\boldsymbol{\alpha}_{n-1}) \tag{2.71}$$

  where η and ϵ are the step-size parameter and the regularization factor, respectively.

- **Acceptance case:** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \delta_0$
  In this case, $\kappa(\cdot, \boldsymbol{u}_n)$ cannot be approximated by the kernel functions already in the dictionary. Then, it will be included into the dictionary and denoted by $\kappa(\cdot, \boldsymbol{u}_{\omega_{M+1}})$. The order M of (2.59) is increased by one, and $\boldsymbol{H}_n$ is updated to p-by-$(M+1)$ matrix. To accommodate the new element in $\boldsymbol{\alpha}_n$, the problem (2.66) is reformulated as

  $$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha} - \boldsymbol{\alpha}_{n-1}\|^2 + \alpha_{M+1}^2 \qquad \text{subject to} \qquad \boldsymbol{d}_n = \boldsymbol{H}_n\boldsymbol{\alpha}. \tag{2.72}$$

  Note that the $(M+1)$-th element α_{M+1} is incorporated to the objective function as a regularizing term. In view of (2.71), we have the following recursion:

  $$\boldsymbol{\alpha}_n = \begin{bmatrix}\boldsymbol{\alpha}_{n-1} \\ 0\end{bmatrix} + \eta\,\boldsymbol{H}_n^\top(\epsilon\boldsymbol{I} + \boldsymbol{H}_n\boldsymbol{H}_n^\top)^{-1}\left(\boldsymbol{d}_n - \boldsymbol{H}_n\begin{bmatrix}\boldsymbol{\alpha}_{n-1} \\ 0\end{bmatrix}\right). \tag{2.73}$$

In summary, we call the set of recursions (2.71) and (2.73) the kernel affine projection (KAP) algorithm. Next, we explore the idea of using instantaneous approximations for the gradient vectors.

2.4.3 Kernel normalized least-mean-square algorithm

Now consider the case $p = 1$. At each time step n, the algorithm described earlier then enforces $d_n = \boldsymbol{h}_n^\top \boldsymbol{\alpha}_n$ where $\boldsymbol{h}_n$ is the column vector whose the m-th entry is $\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})$. The recursive equations (2.72) and (2.73) of KAP algorithm reduce to

- **Rejection case:** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| > \delta_0$

$$\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n-1} + \frac{\eta}{\epsilon + \|\boldsymbol{h}_n\|^2}(d_n - \boldsymbol{h}_n^\top \boldsymbol{\alpha}_{n-1})\boldsymbol{h}_n \tag{2.74}$$

  with $\boldsymbol{h}_n = [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}), \ldots, \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_M})]^\top$.

- **Acceptance case:** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \delta_0$

$$\boldsymbol{\alpha}_n = \begin{bmatrix} \boldsymbol{\alpha}_{n-1} \\ 0 \end{bmatrix} + \frac{\eta}{\epsilon + \|\boldsymbol{h}_n\|^2}\left(d_n - \boldsymbol{h}_n^\top \begin{bmatrix} \boldsymbol{h}_{n-1} \\ 0 \end{bmatrix}\right)\boldsymbol{\kappa}_n \tag{2.75}$$

  with $\boldsymbol{h}_n = [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}), \ldots, \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_{M+1}})]^\top$.

These recursions are referred to as kernel normalized least-mean-square (KNLMS) algorithm. Straightforwardly deleting the normalized terms in the denominator, or based on the framework of LMS algorithm from (2.10) to (2.14), we are able to obtain the well-known kernel least-mean-square (KLMS) algorithm under the CS criterion described as follows:

- **Rejection case:** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| > \delta_0$

$$\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n-1} + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n} \tag{2.76}$$

  with the estimation error $e_n = d_n - \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{\alpha}_{n-1}$ and the kernelized input vector $\boldsymbol{\kappa}_{\omega,n} = [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}), \ldots, \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_M})]^\top$.

- **Acceptance case:** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \delta_0$

$$\boldsymbol{\alpha}_n = \begin{bmatrix} \boldsymbol{\alpha}_{n-1} \\ 0 \end{bmatrix} + \eta\left(d_n - \boldsymbol{\kappa}_{\omega,n}^\top \begin{bmatrix} \boldsymbol{\alpha}_{n-1} \\ 0 \end{bmatrix}\right)\boldsymbol{\kappa}_{\omega,n} \tag{2.77}$$

  with the kernelized input vector $\boldsymbol{\kappa}_{\omega,n} = [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}), \ldots, \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_{M+1}})]^\top$.

It needs to be emphasized that the KLMS algorithm is the starting point for theoretical analysis of kernel adaptive filtering throughout this thesis.

2.4.4 Kernel recursive least-square algorithm

The optimization problem of the kernel recursive least-square (KRLS) algorithm can be formulated as [Honeine 2007]:

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{d}_n - \boldsymbol{H}_n\boldsymbol{\alpha}\|^2 + \epsilon\, \boldsymbol{\alpha}^\top \boldsymbol{K}_n \boldsymbol{\alpha} \tag{2.78}$$

where $\boldsymbol{K}_n$ is the $(M \times M)$ Gram matrix of the elements in online dictionary, and $\boldsymbol{H}_n$ is an $(n \times M)$ matrix with (i, j)-th entry $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_{\omega_j})$. At time step n the solution of the problem (2.78) is

$$\boldsymbol{\alpha}_n = \boldsymbol{P}_n \boldsymbol{H}_n^\top \boldsymbol{d}_n \tag{2.79}$$

with $\boldsymbol{P}_n = (\boldsymbol{H}_n \boldsymbol{H}_n^\top + \epsilon \boldsymbol{K}_n)^{-1}$. Upon the arrival of new input $\boldsymbol{u}_n$, there are also two cases for updating equations of KRLS algorithm using coherence sparsification criterion in the following.

- **Rejection case:** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| > \delta_0$

$$\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n-1} + \frac{\boldsymbol{P}_{n-1}\boldsymbol{\kappa}_{\omega,n}}{1 + \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{P}_{n-1}\boldsymbol{\kappa}_{\omega,n}} (d_n - \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{\alpha}_{n-1}) \tag{2.80}$$

  where

$$\boldsymbol{P}_n = \boldsymbol{P}_{n-1} - \frac{\boldsymbol{P}_{n-1}\boldsymbol{\kappa}_{\omega,n}\boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{P}_{n-1}}{1 + \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{P}_{n-1}\boldsymbol{\kappa}_{\omega,n}} \tag{2.81}$$

  with the kernelized input vector $\boldsymbol{\kappa}_{\omega,n} = [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}), \ldots, \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_M})]^\top$.

- **Acceptance case:** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \delta_0$

  Firstly, the dimension of dictionary becomes $M + 1$. Then, the coefficients vector $\boldsymbol{\alpha}_{n-1}$ and matrix $\boldsymbol{P}_{n-1}$ are computed according to (2.80) and (2.81) to obtain $\tilde{\boldsymbol{\alpha}}_n$ and $\tilde{\boldsymbol{P}}_n$, respectively. The final $\boldsymbol{\alpha}_n$ and $\boldsymbol{P}_n$ are given by

$$\boldsymbol{\alpha}_n = \begin{bmatrix} \tilde{\boldsymbol{\alpha}}_n \\ 0 \end{bmatrix} + \frac{d_n - \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{\alpha}_{n-1}}{1 - \boldsymbol{\kappa}_{\omega,n}^\top \tilde{\boldsymbol{P}}_n \boldsymbol{\kappa}_{\omega,n}} \begin{bmatrix} \boldsymbol{P}_n \boldsymbol{\kappa}_{\omega,n} \\ 1/\kappa(\boldsymbol{u}_n, \boldsymbol{u}_n) \end{bmatrix} \tag{2.82}$$

  where

$$\begin{aligned} \boldsymbol{P}_n = &\begin{bmatrix} \tilde{\boldsymbol{P}}_n & \boldsymbol{0}_{n-1} \\ \boldsymbol{0}_{n-1}^\top & 0 \end{bmatrix} + \frac{1}{1 - \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{P}_n \boldsymbol{\kappa}_{\omega,n}} \begin{bmatrix} -\tilde{\boldsymbol{P}}_n \boldsymbol{\kappa}_{\omega,n} \\ 1/\kappa_k(\boldsymbol{u}_n, \boldsymbol{u}_n) \end{bmatrix} \\ &\times \begin{bmatrix} -(\tilde{\boldsymbol{P}}_n \boldsymbol{\kappa}_{\omega,n})^\top & 1/\kappa_k(\boldsymbol{u}_n, \boldsymbol{u}_n) \end{bmatrix}. \end{aligned} \tag{2.83}$$

The KLMS algorithm converges more slowly than the KRLS algorithm. It however has the advantages of excellent tracking ability and less computational cost relative to the KRLS algorithm. The KAP algorithm has intermediate characteristics between the KRLS and KLMS algorithms in terms of convergence speed, computational complexity and tracking ability.

2.5 Conclusion

This chapter presented some basic aspects of kernel adaptive filtering. We firstly reviewed the classic linear adaptive filtering framework with special emphasis on the LMS algorithm and its theoretical convergence analysis; in the mean of weight vector and in the mean-square sense of weight-error vector. Secondly, we introduced the basic concept and construction of kernel function, which is crucial for kernel-based methods. In addition,

the definition and properties of RKHS involved in kernel adaptive filtering were presented. Thirdly, a brief introduction to sparsification criteria for online dictionary learning, and several types of existing kernel-based adaptive filtering algorithms including KAP, KNLMS, KLMS and KRLS algorithms, were also presented.

In next chapter, we will discuss the stochastic convergence behavior of monokernel LMS algorithm.

Chapter 3

Monokernel LMS algorithm with online dictionary

Contents

3.1 Introduction

Monokernel adaptive filters have been extensively studied over the last decade, and their performances have been investigated experimentally and theoretically on a variety of real-valued nonlinear system identification problems. On the one hand, a very detailed analysis of the stochastic behavior of the KLMS algorithm with single Gaussian kernel was reported in [Parreira 2012], and a closed-form condition for convergence was recently introduced in [Richard 2012]. On the other hand, the QKLMS algorithm with ℓ_1-norm regularization was introduced in [Chen 2012a]. It uses ℓ_1-norm in order to sparsify the parameter vector $\boldsymbol{\alpha}$ in the kernel expansion (2.59) thereby remedying the defect of redundancy of online dictionary. A subgradient approach was considered to accomplish this task,

which contrasts with the more efficient forward-backward splitting algorithm recommended in [Yukawa 2012, Gao 2013].

Except for the above-mentioned works [Yukawa 2012, Chen 2012a], most of the existing strategies for dictionary update are only able to incorporate new elements into the dictionary, and to possibly forget the old ones using a forgetting factor. This means that they cannot automatically discard the obsolete kernel functions, which may be a severe drawback within the context of a time-varying environment. Recently, sparsity-promoting regularization was considered within the context of distributed linear adaptive filtering. All these works propose to use, either the ℓ_1-norm of the vector of filter coefficients as a regularization term, or some other related regularizers to limit the bias relative to the unconstrained solution. The optimization procedures consist of subgradient descent [Chen 2009], projection onto the ℓ_1-ball [Slavakis 2010], or online forward-backward splitting [Murakami 2010]. Surprisingly, this idea was little used in the context of kernel-based adaptive filtering. To the best of our knowledge, only [Yukawa 2012] uses projection for least-squares minimization with weighted block ℓ_1-norm regularization, within the context of multi-kernel adaptive filtering. There is no theoretical work that analyzes the necessity of updating the dictionary in a time-varying environment.

Signal reconstruction from a redundant dictionary has been extensively addressed during the last decade [Candès 2011], both theoretically and experimentally. In order to represent a signal with a minimum number of elements of a dictionary, an efficient approach is to incorporate a sparsity-inducing regularization term such as an ℓ_1-norm one in order to select the most informative patterns. On the other hand, a classical result of adaptive filtering theory says that, as the length of LMS adaptive filters increases, their mean-square estimation error increases and their convergence speed decreases [Haykin 1991]. This intuitively suggests to discard obsolete dictionary elements of monokernel LMS adaptive filters in order to improve their performance in non-stationary environments. To check this property formally, we shall now analyze the behavior of the KLMS algorithm with single Gaussian kernel depicted in [Parreira 2012] in the case where a given proportion of the dictionary elements has distinct stochastic properties from the input samples. No theoretical work has been carried out so far to address this issue. This analytical model will allow us to formally justify the need for updating the dictionary in an online way. It is interesting to note that the generalization presented hereafter was made possible by radically reformulating, and finally simplifying, the mathematical derivation given in [Parreira 2012]. Both models are, however, strictly equivalent in the stationary case, and useful to the theoretical analysis of the multikernel LMS algorithm in next chapter.

In this chapter, we present an analytical study of the convergence behavior of the Gaussian monokernel LMS algorithm in the case where the statistics of the dictionary elements only partially match the statistics of the input data. This analysis highlights the need for updating the dictionary in an online way, by discarding the obsolete elements and adding appropriate ones. Thus, we introduce a monokernel LMS algorithm with ℓ_1-norm regularization in order to automatically perform this task. The stability of this method is analyzed and, finally, its effectiveness is tested with the simulation experiments.

3.1.1 Monokernel LMS algorithms

Several versions of the monokernel LMS algorithm or KLMS algorithm for short in this chapter, whose block diagram is depicted in Figure 3.1, have been proposed in the literature. The two most significant versions consist of considering the problem (2.57) and performing gradient descent on the function $\psi(\cdot)$ in $\mathcal{H}$, or considering the problem (2.58) and performing gradient descent on the parameter vector $\boldsymbol{\alpha}$, respectively. The former strategy is considered in [Liu 2008a, Chen 2012b] for instance, while the latter is applied in [Richard 2009]. Both need the use of an extra mechanism for controlling the order M of the kernel expansion (2.59) at each time instant n. We shall now recall such a model order selection stage, before briefly introducing the parameter update stage we recommend.

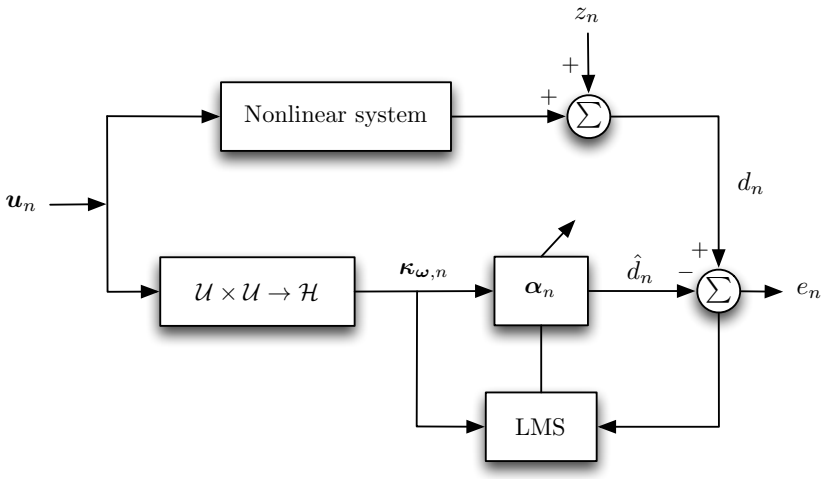


Figure 3.1: Monokernel LMS adaptive filtering.

3.1.1.1 Dictionary update

Recall that the CS criterion described in previous chapter suggests inserting the candidate input $\kappa(\cdot, \boldsymbol{u}_n)$ into the dictionary provided that its coherence remains below a given threshold δ_0

$$\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \delta_0. \tag{3.1}$$

Note that the quantization criterion introduced in [Chen 2012b] consists of comparing $\min_{m=1,\ldots,M} \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_m}\|_2$ with a certain threshold, where $\|\cdot\|_2$ denotes the ℓ_2-norm. It is thus equivalent to the original CS criterion in the case of radial kernels such as the Gaussian one considered hereafter.[1]

[1]Radial kernels are defined as $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = f(\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2)$ with $f \in \mathcal{C}^\infty$ a completely monotonic function on $\mathbb{R}_+$, i.e., the k-th derivative of f satisfies $(-1)^k f^{(k)}(r) \geq 0$ for all $r \in \mathbb{R}_+^*$, $k \geq 0$. See [Cucker 2001]. Decreasing of f on $\mathbb{R}_+$ ensures the equivalence between the coherence criterion and the quantization criterion.

3.1.1.2 Filter parameter update

At iteration n, upon the arrival of new data $(\boldsymbol{u}_n, d_n)$, one of the following alternatives holds. If $\kappa(\cdot, \boldsymbol{u}_n)$ does not satisfy the coherence rule (3.1), the dictionary remains unaltered. On the other hand, if condition (3.1) is met, the kernel function $\kappa(\cdot, \boldsymbol{u}_n)$ is inserted into the dictionary where it is then denoted by $\kappa(\cdot, \boldsymbol{u}_{\omega_{M+1}})$. The least-mean-square algorithm applied to the parametric form (2.58) leads to the KLMS algorithm with single Gaussian kernel function as presented in Chapter 2. For simplicity, note that we have voluntarily omitted the regularization term in (2.58), that is, $\gamma = 0$.

- **Rejection case:** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| > \delta_0$

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n} \tag{3.2}$$

  with η the step-size parameter. The estimation error is $e_n = d_n - \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{\alpha}_n$ with $\boldsymbol{\kappa}_{\omega,n} = [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}), \cdots, \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_M})]^\top$.

- **Acceptance case:** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \delta_0$

$$\boldsymbol{\alpha}_{n+1} = \begin{bmatrix} \boldsymbol{\alpha}_n \\ 0 \end{bmatrix} + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n} \tag{3.3}$$

  with the kernelized input vector $\boldsymbol{\kappa}_{\omega,n} = [\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}), \cdots, \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_{M+1}})]^\top$.

The CS criterion guarantees that the dictionary dimension is finite for any input sequence $\{\boldsymbol{u}_n\}_{n=1}^{\infty}$ due to the compactness of the input space $\mathcal{U}$ [Richard 2009, proposition 2].

3.2 Mean square error analysis

Consider the nonlinear system identification problem shown in Figure 3.1, and the finite-order model (2.59) based on the Gaussian kernel function $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp\left(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2 / 2\xi^2\right)$. The order M of the model (2.59) or, equivalently, the size M of the dictionary $\mathcal{D}$, is assumed known and fixed throughout the analysis. The nonlinear system input data $\boldsymbol{u}_n \in \mathbb{R}^{q\times 1}$ are supposed to be zero-mean, independent, and identically distributed Gaussian vectors. We consider that the entries of $\boldsymbol{u}_n$ can be correlated, and we denote by $\boldsymbol{R}_{\boldsymbol{u}\boldsymbol{u}} = E\{\boldsymbol{u}_n \boldsymbol{u}_n^\top\}$ the autocorrelation matrix of the input data. It is assumed that the input data $\boldsymbol{u}_n$ or the transformed inputs by kernel $\psi(\boldsymbol{u}_n)$ are locally or temporally stationary in the environment needed to be analyzed. The estimated system output is given by

$$\hat{d}_n = \boldsymbol{\alpha}_n^\top \boldsymbol{\kappa}_{\omega,n} \tag{3.4}$$

with $\boldsymbol{\alpha}_n = [\alpha_1(n), \ldots, \alpha_M(n)]^\top$. The corresponding estimation error is defined as

$$e_n = d_n - \hat{d}_n. \tag{3.5}$$

Squaring both sides of (3.5) and taking the expected value leads to the MSE

$$J_{\text{ms}}(n) = E\{e_n^2\} = E\{d_n^2\} - 2\,\boldsymbol{p}_{\kappa d}^\top \boldsymbol{\alpha}_n + \boldsymbol{\alpha}_n^\top \boldsymbol{R}_{\kappa\kappa}\, \boldsymbol{\alpha}_n \tag{3.6}$$

where $\boldsymbol{R}_{\kappa\kappa} = E\{\boldsymbol{\kappa}_{\omega,n}\boldsymbol{\kappa}_{\omega,n}^{\top}\}$ is the correlation matrix of the kernelized input $\boldsymbol{\kappa}_{\omega,n}$, and $\boldsymbol{p}_{\kappa d} = E\{d_n\,\boldsymbol{\kappa}_{\omega,n}\}$ is the cross-correlation vector between $\boldsymbol{\kappa}_{\omega,n}$ and d_n. It has already been proved that $\boldsymbol{R}_{\kappa\kappa}$ is positive definite [Parreira 2012] if the input data $\boldsymbol{u}_n$ are independent and identically distributed Gaussian vectors, and, as a consequence, the dictionary elements $\boldsymbol{u}_{\omega_i}$ and $\boldsymbol{u}_{\omega_j}$ are statistically independent for $i \neq j$. Thus, the optimum weight vector is given by

$$\boldsymbol{\alpha}_{\text{opt}} = \boldsymbol{R}_{\kappa\kappa}^{-1}\,\boldsymbol{p}_{\kappa d} \tag{3.7}$$

and the corresponding minimum MSE is

$$J_{\min} = E\{d_n^2\} - \boldsymbol{p}_{\kappa d}^{\top}\,\boldsymbol{R}_{\kappa\kappa}^{-1}\,\boldsymbol{p}_{\kappa d}. \tag{3.8}$$

Note that expressions (3.7) and (3.8) are the well-known Wiener solution and minimum MSE, [Sayed 2003, Haykin 1991], respectively, where the input signal vector has been replaced by the kernelized input vector.

In order to determine $\boldsymbol{\alpha}_{\text{opt}}$, we shall now calculate the correlation matrix $\boldsymbol{R}_{\kappa\kappa}$ using the statistical properties of the input $\boldsymbol{u}_n$ and the kernel definition. Let us introduce the following notations

$$\|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_i}\|_2^2 + \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_j}\|_2^2 = \boldsymbol{y}_3^{\top}\,\boldsymbol{Q}_3\,\boldsymbol{y}_3 \tag{3.9}$$

with

$$\boldsymbol{y}_3 = \begin{pmatrix}\boldsymbol{u}_n^{\top} & \boldsymbol{u}_{\omega_i}^{\top} & \boldsymbol{u}_{\omega_j}^{\top}\end{pmatrix}^{\top} \tag{3.10}$$

and

$$\boldsymbol{Q}_3 = \begin{pmatrix} 2\boldsymbol{I} & -\boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} & \boldsymbol{O} \\ -\boldsymbol{I} & \boldsymbol{O} & \boldsymbol{I} \end{pmatrix} \tag{3.11}$$

where $\boldsymbol{I}$ is the $(q \times q)$ identity matrix, and $\boldsymbol{O}$ is the $(q \times q)$ null matrix. From [Omura 1965, p. 100], we know that the moment generating function of a quadratic form $z = \boldsymbol{y}^{\top}\boldsymbol{Q}\,\boldsymbol{y}$, where $\boldsymbol{y}$ is a zero-mean Gaussian vector with covariance $\boldsymbol{R}_y$, is given by

$$\psi_z(s) = E\{e^{sz}\} = \left[\det\{\boldsymbol{I} - 2s\boldsymbol{Q}\boldsymbol{R}_y\}\right]^{-1/2}. \tag{3.12}$$

Making $s = -1/(2\xi^2)$ in equation (3.12), we find that the (i,j)-th element of $\boldsymbol{R}_{\kappa\kappa}$ is given by

$$[\boldsymbol{R}_{\kappa\kappa}]_{ij} = \begin{cases} r_{\text{md}} = \left[\det\left\{\boldsymbol{I}_3 + \boldsymbol{Q}_3\,\boldsymbol{R}_3(i,j)/\xi^2\right\}\right]^{-1/2}, & i = j \\ r_{\text{od}} = \left[\det\left\{\boldsymbol{I}_3 + \boldsymbol{Q}_3\,\boldsymbol{R}_3(i,j)/\xi^2\right\}\right]^{-1/2}, & i \neq j \end{cases} \tag{3.13}$$

with $1 \leq i,\, j \leq M$, and r_{md} and r_{od} are the main-diagonal and off-diagonal entries of $\boldsymbol{R}_{\kappa\kappa}$, respectively. In equation (3.13), $\boldsymbol{R}_p$ is the $(pq \times pq)$ correlation matrix of vector $\boldsymbol{y}_p$ (see expression (3.14) for an illustration of this notation with $p = 3$), $\boldsymbol{I}_p$ is the $(pq \times pq)$ identity matrix, and $\det\{\cdot\}$ denotes the determinant of a matrix. The two cases $(i = j)$ and $(i \neq j)$ correspond to different forms of the $(3q \times 3q)$ matrix $\boldsymbol{R}_3(i,j)$, given by

$$\boldsymbol{R}_3(i,j) = \begin{pmatrix} \boldsymbol{R}_{\boldsymbol{u}\boldsymbol{u}} & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,i) & \boldsymbol{R}_{\mathcal{D}}(i,j) \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,j) & \boldsymbol{R}_{\mathcal{D}}(j,j) \end{pmatrix} \tag{3.14}$$

where $\boldsymbol{R}_{\mathcal{D}}(i,j) = E\{\boldsymbol{u}_{\omega_i}\boldsymbol{u}_{\omega_j}^{\top}\}$ is the intercorrelation matrix of the dictionary elements. Compared with [Parreira 2012], the formulations (3.13)–(3.14), and other reformulations pointed out in the following, allow to address more general problems by making the analyses tractable. In particular, in order to evaluate the effects of a mismatch between the input data and the dictionary elements, we shall now consider the case where that they do not necessarily share the same statistical properties. This situation will occur in a time-varying environment with most of the existing dictionary update strategies. Indeed, they are only able to incorporate new elements into the dictionary, and cannot automatically discard obsolete ones. In [Yukawa 2012, Chen 2012a, Gao 2013], it is suggested to use a sparsity-promoting ℓ_1-norm regularization term to allow minor contributors in the kernel dictionary to be automatically discarded, both without theoretical results. However, on the one hand, the algorithm [Yukawa 2012] was proposed in the multi-kernel context. On the other hand, the algorithm [Chen 2012a] uses a subgradient approach and has quadratic computational complexity in M.

We shall now suppose that the first L dictionary elements $\{\boldsymbol{u}_{\omega_m} \in \mathbb{R}^q : 1 \leq m \leq L\}$ have the same autocorrelation matrix $\boldsymbol{R}_{\boldsymbol{uu}}$ as the input $\boldsymbol{u}_n$, whereas the other $(M-L)$ elements $\{\boldsymbol{u}_{\omega_m} \in \mathbb{R}^q : L < m \leq M\}$ have a distinct autocorrelation matrix denoted by $\tilde{\boldsymbol{R}}_{\boldsymbol{uu}}$. In this case, $\boldsymbol{R}_{\mathcal{D}}(i,j)$ in equation (3.14) is given by

$$\boldsymbol{R}_{\mathcal{D}}(i,j) = \begin{cases} \boldsymbol{R}_{\boldsymbol{uu}}, & 1 \leq i = j \leq L \\ \widetilde{\boldsymbol{R}}_{\boldsymbol{uu}}, & L < i = j \leq M \\ \boldsymbol{O}, & 1 \leq i \neq j \leq M \end{cases} \tag{3.15}$$

which allows to calculate the correlation matrix $\boldsymbol{R}_{\kappa\kappa}$ of the kernelized input via equation (3.13). Note that $\boldsymbol{R}_{\mathcal{D}}(i,j)$ in equation (3.14) reduces to $\delta_{ij}\,\boldsymbol{R}_{\boldsymbol{uu}}$, with $\delta_{ij} = 1$ if $(i = j)$, otherwise 0, in the case $(L = M)$ considered in [Parreira 2012].

3.3 Transient behavior analysis

We shall now analyze the transient behavior of the algorithm. We shall successively focus on the convergence of the weight vector in the mean sense, i.e., $E\{\boldsymbol{\alpha}_n\}$, and of the mean square error $J_{\text{ms}}(n)$ defined in (3.6).

3.3.1 Mean weight behavior

The weight update equation of KLMS algorithm is given by

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n}. \tag{3.16}$$

By defining the weight error vector $\boldsymbol{v}_n = \boldsymbol{\alpha}_n - \boldsymbol{\alpha}_{\text{opt}}$ and subtracting $\boldsymbol{\alpha}_{\text{opt}}$ from both side of (3.16), leads to the weight error vector update equation

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n}. \tag{3.17}$$

From (3.4) and (3.5), and the definition of $\boldsymbol{v}_n$, the error equation is given by

$$e_n = d_n - \boldsymbol{\kappa}_{\omega,n}^{\top}\,\boldsymbol{v}_n - \boldsymbol{\kappa}_{\omega,n}^{\top}\,\boldsymbol{\alpha}_{\text{opt}} \tag{3.18}$$

and the optimal estimation error is

$$e_n^o = d_n - \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{\alpha}_{\text{opt}}. \tag{3.19}$$

Substituting (3.18) into (3.17) yields

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \eta\, d_n\, \boldsymbol{\kappa}_{\omega,n} - \eta\, \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{v}_n\, \boldsymbol{\kappa}_{\omega,n} - \eta\, \boldsymbol{\kappa}_{\omega,n}^\top \boldsymbol{\alpha}_{\text{opt}}\, \boldsymbol{\kappa}_{\omega,n}. \tag{3.20}$$

Simplifying assumptions are required in order to make the study of the stochastic behavior of $\boldsymbol{\kappa}_{\omega,n}$ mathematically feasible. The so-called modified independence assumption (MIA) suggests that $\boldsymbol{\kappa}_{\omega,n}\boldsymbol{\kappa}_{\omega,n}^\top$ is statistically independent of $\boldsymbol{v}_n$. It is justified in detail in [Minkoff 2001], and shown to be less restrictive than the independence assumption [Sayed 2003]. We also assume that the finite-order model provides a close enough approximation to the infinite-order model with minimum MSE, so that $E\{e_n^o\} \approx 0$. Taking the expected value of both sides of equation (3.20) and using these two assumptions yields

$$E\{\boldsymbol{v}_{n+1}\} = (\boldsymbol{I} - \eta\, \boldsymbol{R}_{\kappa\kappa})\, E\{\boldsymbol{v}_n\}. \tag{3.21}$$

This expression corresponds to the LMS mean weight behavior for the kernelized input vector $\boldsymbol{\kappa}_{\omega,n}$.

3.3.2 Mean square error behavior

Using equation (3.18) and the MIA, the second-order moments of the weights are related to the MSE through [Sayed 2003]

$$J_{\text{ms}}(n) = J_{\min} + \text{trace}\{\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_v(n)\} \tag{3.22}$$

where $\boldsymbol{C}_v(n) = E\{\boldsymbol{v}_n\boldsymbol{v}_n^\top\}$ is the autocorrelation matrix of the weight error vector $\boldsymbol{v}_n$, $J_{\min} = E\{e_n^{o\,2}\}$ denotes the minimum MSE, and $\text{trace}\{\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_v(n)\}$ is the excess MSE (EMSE). The analysis of the MSE behavior (3.22) requires a model for $\boldsymbol{C}_v(n)$, which is highly affected by the kernelization of the input signal $\boldsymbol{u}_n$. An analytical model for the behavior of $\boldsymbol{C}_v(n)$ was derived in [Parreira 2012]. Using simplifying assumptions derived from the MIA, it reduces to the following recursion

$$\boldsymbol{C}_v(n+1) \approx \boldsymbol{C}_v(n) - \eta\,[\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_v(n) + \boldsymbol{C}_v(n)\boldsymbol{R}_{\kappa\kappa}] + \eta^2\,\boldsymbol{T}(n) + \eta^2\,\boldsymbol{R}_{\kappa\kappa}J_{\min} \tag{3.23a}$$

with

$$\boldsymbol{T}(n) = E\{\boldsymbol{\kappa}_{\omega,n}\,\boldsymbol{\kappa}_{\omega,n}^\top\,\boldsymbol{v}_n\,\boldsymbol{v}_n^\top\,\boldsymbol{\kappa}_{\omega,n}\,\boldsymbol{\kappa}_{\omega,n}^\top\}. \tag{3.23b}$$

The evaluation of expectation (3.23b) is an important step in the analysis. It leads to extensive calculus if proceeding as in [Parreira 2012] because, as $\boldsymbol{\kappa}_{\omega,n}$ is a nonlinear transformation of a quadratic function of the Gaussian input vector $\boldsymbol{u}_n$, it is neither zero-mean nor Gaussian. In this chapter, we provide an equivalent approach that greatly simplifies the calculation. This allows us to consider the general case where there is possibly a mismatch between the statistics of the input data $\boldsymbol{u}_n$ and the dictionary elements. Using the MIA to determine the (i,j)-th element of $\boldsymbol{T}(n)$ in equation (3.23b) yields

$$[\boldsymbol{T}(n)]_{ij} \approx \sum_{\ell=1}^{M}\sum_{p=1}^{M} E\{\kappa_{\omega,n}(i)\,\kappa_{\omega,n}(j)\,\kappa_{\omega,n}(\ell)\,\kappa_{\omega,n}(p)\} \cdot [\boldsymbol{C}_v(n)]_{\ell p} \tag{3.24}$$

where $\kappa_{\omega,n}(i) = \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_i})$. This expression can be written as

$$[\boldsymbol{T}(n)]_{ij} \approx \text{trace}\{\boldsymbol{K}(i,j)\,\boldsymbol{C}_v(n)\} \tag{3.25}$$

where the (ℓ, p)-th entry of $\boldsymbol{K}(i,j)$ is given by $[\boldsymbol{K}(i,j)]_{\ell,p} = E\{e^{sz}\}$, with $s = -1/(2\xi^2)$ and

$$z = \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_i}\|_2^2 + \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_j}\|_2^2 + \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_\ell}\|_2^2 + \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_p}\|_2^2. \tag{3.26}$$

Using expression (3.12) leads us to

$$[\boldsymbol{K}(i,j)]_{\ell,p} = \left[\det\{\boldsymbol{I}_5 + \boldsymbol{Q}_5\,\boldsymbol{R}_5(i,j,\ell,p)/\xi^2\}\right]^{-1/2} \tag{3.27}$$

with

$$\boldsymbol{Q}_5 = \begin{pmatrix} 4\boldsymbol{I} & -\boldsymbol{I} & -\boldsymbol{I} & -\boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} \\ -\boldsymbol{I} & \boldsymbol{O} & \boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} \\ -\boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{I} & \boldsymbol{O} \\ -\boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{I} \end{pmatrix}, \tag{3.28}$$

and

$$\boldsymbol{R}_5(i,j,\ell,p) = \begin{pmatrix} \boldsymbol{R}_{\boldsymbol{uu}} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,i) & \boldsymbol{R}_{\mathcal{D}}(i,j) & \boldsymbol{R}_{\mathcal{D}}(i,\ell) & \boldsymbol{R}_{\mathcal{D}}(i,p) \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,j) & \boldsymbol{R}_{\mathcal{D}}(j,j) & \boldsymbol{R}_{\mathcal{D}}(j,\ell) & \boldsymbol{R}_{\mathcal{D}}(j,p) \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,\ell) & \boldsymbol{R}_{\mathcal{D}}(j,\ell) & \boldsymbol{R}_{\mathcal{D}}(\ell,\ell) & \boldsymbol{R}_{\mathcal{D}}(\ell,p) \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,p) & \boldsymbol{R}_{\mathcal{D}}(j,p) & \boldsymbol{R}_{\mathcal{D}}(\ell,p) & \boldsymbol{R}_{\mathcal{D}}(p,p) \end{pmatrix}, \tag{3.29}$$

which uses the same block definition as in (3.15). Again, note that $\boldsymbol{R}_{\mathcal{D}}(i,j)$ in the above equation reduces to $\delta_{ij}\,\boldsymbol{R}_{\boldsymbol{uu}}$ in the regular case ($L = M$) considered in [Parreira 2012]. This expression concludes the calculation.

3.4 Steady-state behavior

We shall now determine the steady-state of the recursion (3.23a). Observing that it only involves linear operations on the entries of $\boldsymbol{C}_v(n)$, we can rewrite this equation in a vectorial form in order to simplify the derivations. Let $\text{vec}\{\cdot\}$ denote the operator that stacks the columns of a matrix on top of each other. Vectorizing the matrices $\boldsymbol{C}_v(n)$ and $\boldsymbol{R}_{\kappa\kappa}$ by $\boldsymbol{c}_v(n) = \text{vec}\{\boldsymbol{C}_v(n)\}$ and $\boldsymbol{r}_{\kappa\kappa} = \text{vec}\{\boldsymbol{R}_{\kappa\kappa}\}$, it can be verified that (3.23a) can be rewritten as

$$\boldsymbol{c}_v(n+1) = \boldsymbol{G}\,\boldsymbol{c}_v(n) + \eta^2 J_{\min}\,\boldsymbol{r}_{\kappa\kappa} \tag{3.30}$$

with

$$\boldsymbol{G} = \boldsymbol{I} - \eta(\boldsymbol{G}_1 + \boldsymbol{G}_2) + \eta^2\boldsymbol{G}_3. \tag{3.31}$$

Matrix $\boldsymbol{G}$ is found by the use of the following definitions:

- $\boldsymbol{I}$ is the identity matrix of dimension $M^2 \times M^2$.

- $\boldsymbol{G}_1$ is involved in the product $\boldsymbol{C}_v(n)\boldsymbol{R}_{\kappa\kappa}$, and is given by $\boldsymbol{G}_1 = \boldsymbol{I} \otimes \boldsymbol{R}_{\kappa\kappa}$. It is thus a block-diagonal matrix, with $\boldsymbol{R}_{\kappa\kappa}$ on its diagonal, where $\otimes$ denotes the Kronecker tensor product.
- $\boldsymbol{G}_2$ is involved in the product $\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_v(n)$, and can be written as $\boldsymbol{R}_{\kappa\kappa} \otimes \boldsymbol{I}$.
- $\boldsymbol{G}_3$ is involved in the matrix $\boldsymbol{T}(n)$ defined as in equation (3.25), namely,

$$[\boldsymbol{G}_3]_{i+(j-1)M,\ell+(p-1)M} = [\boldsymbol{K}(i,j)]_{\ell,p} \tag{3.32}$$

  with $1 \leq i, j, \ell, p \leq M$.

Note that $\boldsymbol{G}_1$ to $\boldsymbol{G}_3$ are symmetric matrices, which implies that $\boldsymbol{G}$ is also symmetric. Assuming convergence, the closed-formed solution of the recursion (3.30) is given by

$$\boldsymbol{c}_v(n) = \boldsymbol{G}^n \left[\boldsymbol{c}_v(0) - \boldsymbol{c}_v(\infty)\right] + \boldsymbol{c}_v(\infty) \tag{3.33}$$

where $\boldsymbol{c}_v(\infty)$ denotes the vector $\boldsymbol{c}_v(n)$ in steady-state, which is given by

$$\boldsymbol{c}_v(\infty) = \eta^2 \, J_{\min} \, (\boldsymbol{I} - \boldsymbol{G})^{-1} \, \boldsymbol{r}_{\kappa\kappa}. \tag{3.34}$$

From equation (3.22), the steady-state MSE is finally given by

$$J_{\text{ms}}(\infty) = J_{\min} + \text{trace}\{\boldsymbol{R}_{\kappa\kappa} \, \boldsymbol{C}_v(\infty)\} \tag{3.35}$$

where $J_{\text{ex}}(\infty) = \text{trace}\{\boldsymbol{R}_{\kappa\kappa} \, \boldsymbol{C}_v(\infty)\}$ is the steady-state EMSE.

3.5 Simulation results and discussion

In this section, we present simulation examples to illustrate the accuracy of the derived model, and to study the properties of the algorithm in the case where the statistics of the dictionary elements partially match the statistics of the input data. This first experiment provides the motivation for deriving the online dictionary learning algorithm described subsequently, which can automatically discard the obsolete elements and add appropriate ones.

Two examples with abrupt variance changes in the input signal are presented hereafter. In each situation, the size of the dictionary was fixed beforehand, and the entries of the dictionary elements were i.i.d. randomly generated from a zero-mean Gaussian distribution. Each time series was divided into two subsequences. For the first one, the variance of this distribution was set as equal to the variance of the input signal. For the second one, it was abruptly set to a smaller or larger value in order to simulate a dictionary misadjustment. All the parameters were chosen within reasonable ranges collected from the literature.

Notation: In Tables 3.1 and 3.2, dictionary settings are compactly expressed as $\mathcal{D}_i = \{M_i@\sigma_i\} \cup \{M_i'@\sigma_i'\}$. This has to be interpreted as: Dictionary $\mathcal{D}_i$ is composed of M_i vectors with entries i.i.d. randomly generated from a zero-mean Gaussian distribution with standard deviation σ_i, and M_i' vectors with entries i.i.d. randomly generated from a zero-mean Gaussian distribution with standard deviation σ_i'.

3.5.1 Example 1

Consider the problem studied in [Parreira 2012, Narendra 1990, Mandic 2004], for which

$$\begin{cases} y_n = \dfrac{y_{n-1}}{1+y_{n-1}^2} + u_{n-1}^3 \\ d_n = y_n + z_n \end{cases} \tag{3.36}$$

where the output signal y_n was corrupted by a zero-mean i.i.d. Gaussian noise z_n with variance $\sigma_z^2 = 10^{-4}$. The input sequence u_n was i.i.d. randomly generated from a zero-mean Gaussian distribution with two possible standard deviations, $\sigma_u = 0.35$ or 0.15, to simulate an abrupt change between two subsequences. The overall length of the input sequence was 4×10^4. Distinct dictionaries, denoted by $\mathcal{D}_1$ and $\mathcal{D}_2$, were used for each subsequence. The Gaussian kernel bandwidth ξ was set to 0.02, and the KLMS step-size η was set to 0.01. Two situations were investigated. For the first one, the standard deviation of the input signal was changed from 0.35 to 0.15 at time instant $n = 2 \times 10^4$. Conversely, in the second one, it was changed from 0.15 to 0.35.

Table 3.1 presents the simulation conditions, and the experimental results based on 200 Monte Carlo runs. The convergence iteration number n_ϵ was determined in order to satisfy

$$\|\boldsymbol{c}_v(\infty) - \boldsymbol{c}_v(n_\epsilon)\|_2 \leq 10^{-3}. \tag{3.37}$$

Note that $J_{\min}$, $J_{\text{ms}}(\infty)$, $J_{\text{ex}}(\infty)$ and n_ϵ concern convergence in the second subsequence, with the dictionary $\mathcal{D}_2$. The learning curves are depicted in Figures 3.2 and 3.3.

Table 3.1: Summary of simulation results for Example 1.

ξ	η	σ_u	$\mathcal{D}_1$	$\mathcal{D}_2$	$J_{\min}$ [dB]	$J_{\text{ms}}(\infty)$ [dB]	$J_{\text{ex}}(\infty)$ [dB]	n_ϵ
0.02	0.01	$0.35 \rightarrow 0.15$	{10@0.35}	{10@0.35}	-22.04	-22.03	-49.33	32032
				{10@0.15}	**-22.50**	**-22.49**	-47.25	26538
				{10@0.15} ∪ {10@0.35}	-21.90	-21.87	-44.71	30889
0.02	0.01	$0.15 \rightarrow 0.35$	{10@0.15}	{10@0.15}	-10.98	-10.97	-38.26	32509
				{10@0.35}	**-11.20**	**-11.19**	-39.64	36061
				{10@0.15} ∪ {10@0.35}	-11.01	-10.99	-35.81	31614

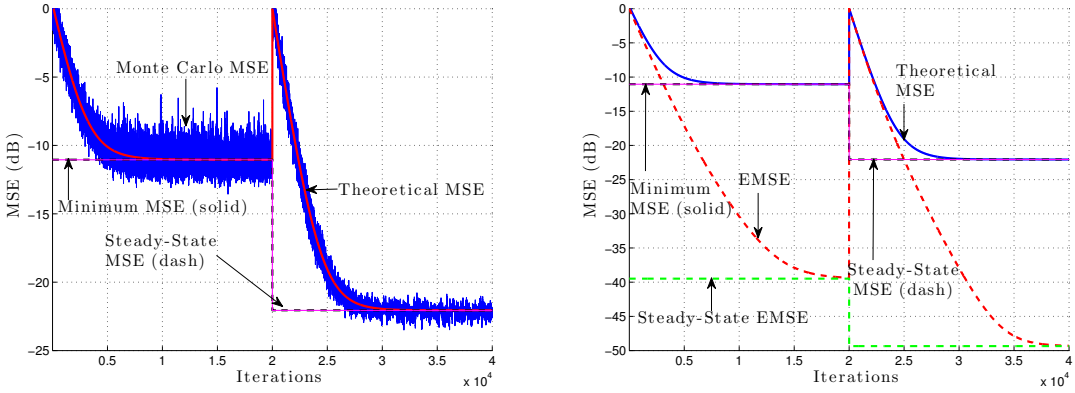
3.5.2 Example 2

Consider the nonlinear dynamic system studied in [Parreira 2012, Vörös 2003] where the input signal was a sequence of statistically independent vectors

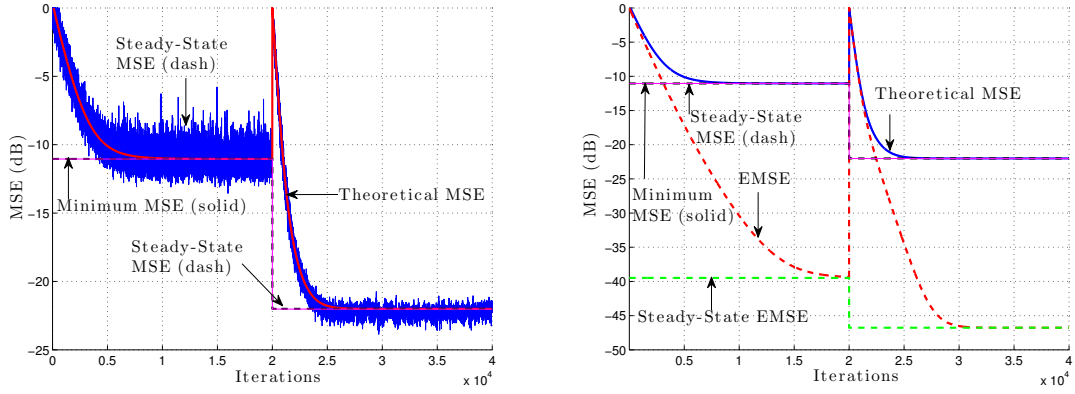
$$\boldsymbol{u}_n = [u_1(n)\ u_2(n)]^\top \tag{3.38}$$

with correlated samples satisfying $u_1(n) = 0.5\, u_2(n) + v_u(n)$. The second component of $\boldsymbol{u}_n$, and $v_u(n)$, were i.i.d. zero-mean Gaussian sequences with standard deviation both equal to $\sqrt{0.0656}$, or to $\sqrt{0.0156}$, during the two subsequences of input data. We considered the linear system with memory defined by

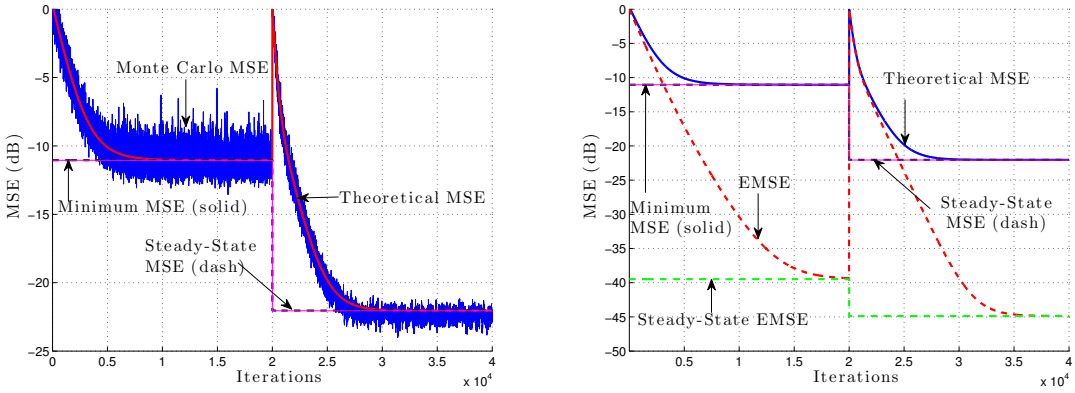
$$y_n = \boldsymbol{a}^\top \boldsymbol{u}_n - 0.2\, y_{n-1} + 0.35\, y_{n-2} \tag{3.39}$$



(a) $\mathcal{D}_2 = \{10@0.35\}$

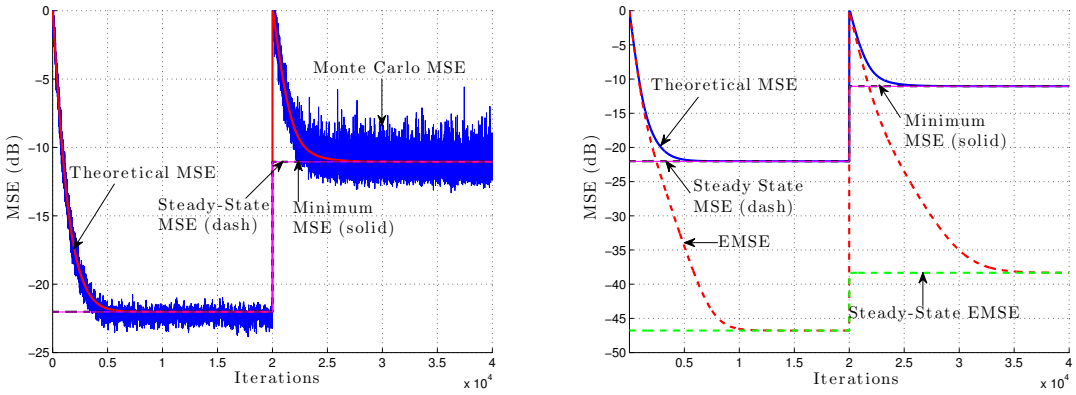


(b) $\mathcal{D}_2 = \{10@0.15\}$

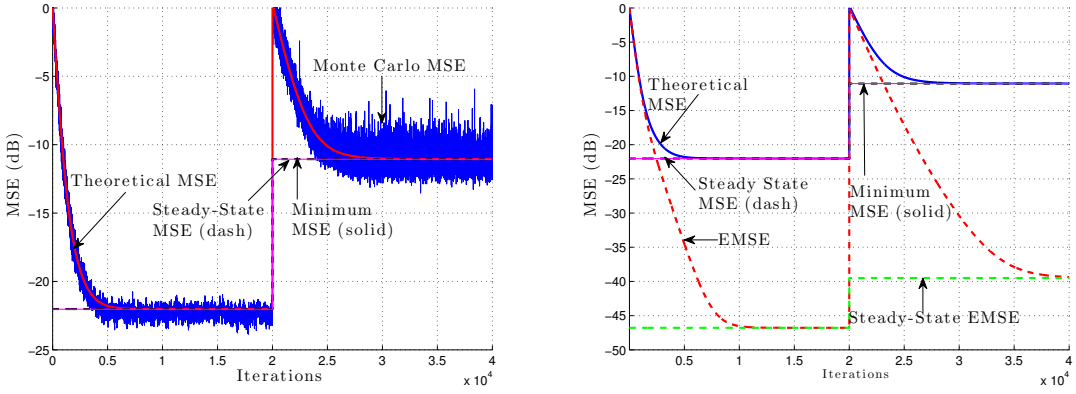


(c) $\mathcal{D}_2 = \{10@0.15\} \cup \{10@0.35\}$

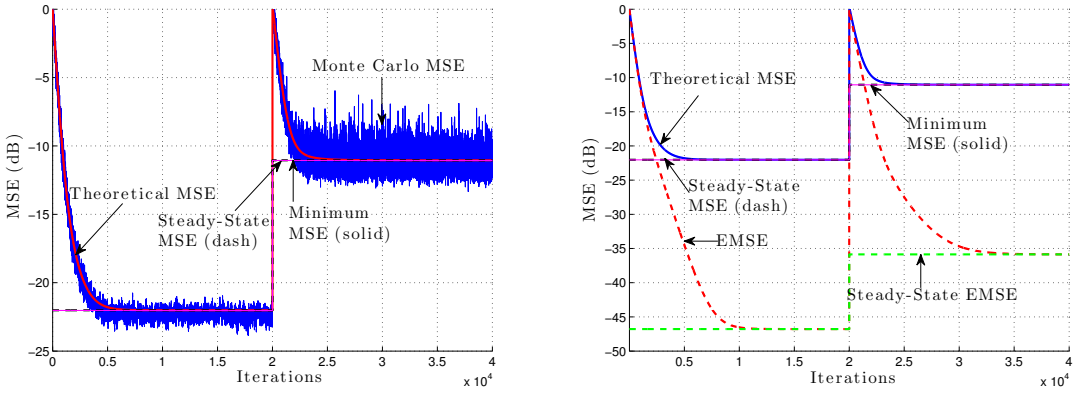
Figure 3.2: Learning curves for Example 1 where $\sigma_u : 0.35 \rightarrow 0.15$ and $\mathcal{D}_1 = \{10@0.35\}$. See the first row of Table 3.1.



(a) $\mathcal{D}_2 = \{10@0.15\}$



(b) $\mathcal{D}_2 = \{10@0.35\}$



(c) $\mathcal{D}_2 = \{10@0.15\} \cup \{10@0.35\}$

Figure 3.3: Learning curves for Example 1 where $\sigma_u : 0.15 \to 0.35$ and $\mathcal{D}_1 = \{10@0.15\}$. See the second row of Table 3.1.

where $\boldsymbol{a} = [1\ 0.5]^\top$ and a nonlinear Wiener function

$$\varphi(y_n) = \begin{cases} \dfrac{y_n}{3[0.1 + 0.9\, y_n^2]^{1/2}} & \text{for } y_n \geq 0 \\ \dfrac{-y_n^2[1 - \exp(0.7y_n)]}{3} & \text{for } y_n < 0 \end{cases} \tag{3.40}$$

$$d_n = \varphi(y_n) + z_n \tag{3.41}$$

where d_n is the output signal. It was corrupted by a zero-mean i.i.d. Gaussian noise z_n with variance $\sigma_z^2 = 10^{-6}$. The initial condition $y_1 = 0$ was considered. The bandwidth ξ of the Gaussian kernel was set to 0.05, and the step-size η of the KLMS was set to 0.05. The length of each input sequence was 4×10^4. As in Example 1, two changes were considered. For the first one, the standard deviation of $u_2(n)$ and $v_u(n)$ was changed from $\sqrt{0.0656}$ to $\sqrt{0.0156}$ at time instant $n = 1 \times 10^4$. Conversely, for the second one, it was changed from $\sqrt{0.0156}$ to $\sqrt{0.0656}$.

Table 3.2 presents the results based on 200 Monte Carlo runs. Note that $J_{\min}$, $J_{\text{ms}}(\infty)$, $J_{\text{ex}}(\infty)$ and n_ϵ concern convergence in the second subsequence, with dictionary $\mathcal{D}_2$. The learning curves are depicted in Figures 3.4 and 3.5.

Table 3.2: Summary of simulation results for Example 2.

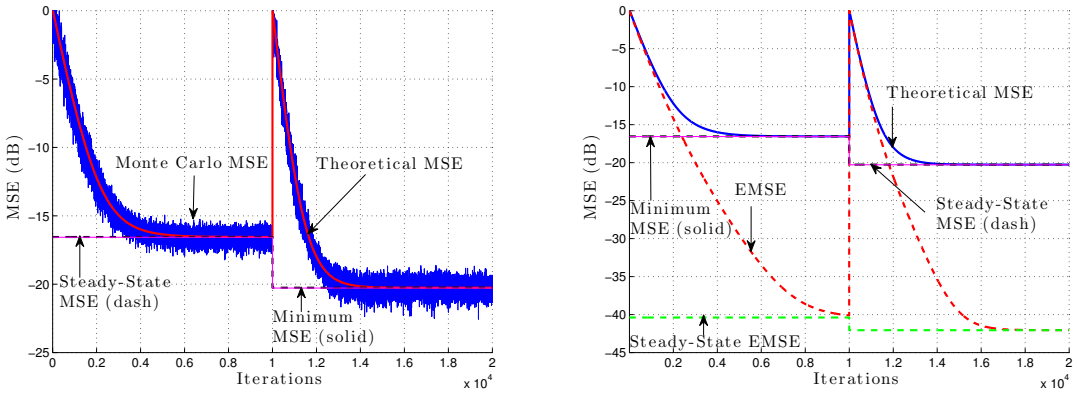
ξ	η	$\sigma_{u_2}, \sigma_{v_u}$	$\mathcal{D}_1$	$\mathcal{D}_2$	$J_{\min}$ [dB]	$J_{\text{ms}}(\infty)$ [dB]	$J_{\text{ex}}(\infty)$ [dB]	n_ϵ
0.05	0.05	$\sqrt{0.0656} \to \sqrt{0.0156}$	$\{15@\sqrt{0.0656}\}$	$\{15@\sqrt{0.0656}\}$	-20.28	-20.25	-42.04	15519
				$\{15@\sqrt{0.0156}\}$	-20.27	-20.20	-37.96	12117
				$\{15@\sqrt{0.0156}\} \cup \{15@\sqrt{0.0656}\}$	**-20.47**	**-20.37**	-36.68	14731
0.05	0.05	$\sqrt{0.0156} \to \sqrt{0.0656}$	$\{15@\sqrt{0.0156}\}$	$\{15@\sqrt{0.0156}\}$	-16.40	-16.37	-38.12	15858
				$\{15@\sqrt{0.0656}\}$	-16.57	-16.55	-40.39	19269
				$\{15@\sqrt{0.0156}\} \cup \{15@\sqrt{0.0656}\}$	**-16.61**	**-16.57**	-36.21	16123

3.5.3 Discussion

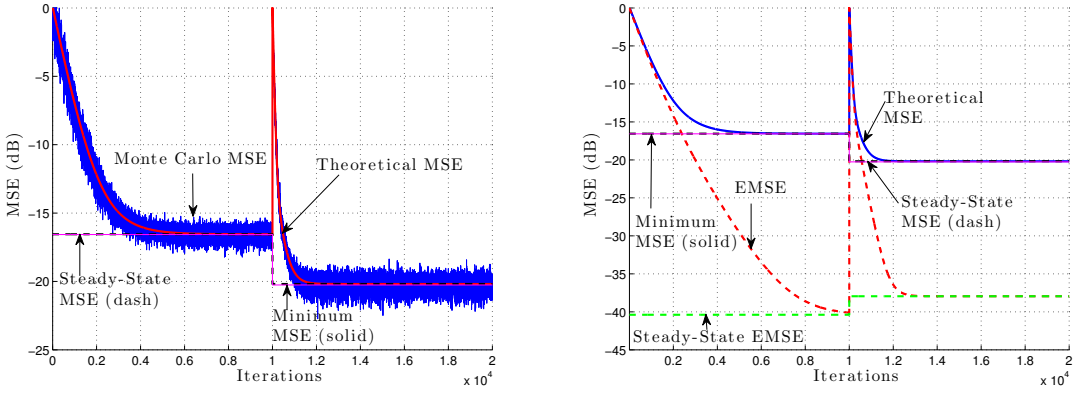
We shall now discuss the simulation results. It is important to recognize the significance of the mean-square estimation errors provided by the model, which perfectly match the averaged Monte Carlo simulation results. The model separates the contribution of the minimum MSE and EMSE, and makes comparisons possible. The simulation results clearly show that adjusting the dictionary to the input signal has a positive effect on the performance when a change in the statistics is detected. This can be done by adding new elements to the existing dictionary, while at the same time possibly discarding the obsolete elements. Considering a completely new dictionary led us to the lowest MSE $J_{\text{ms}}(\infty)$ and minimum MSE $J_{\min}$ in Example 1. Adding new elements to the existing dictionary provided the lowest MSE $J_{\text{ms}}(\infty)$ and minimum MSE $J_{\min}$ in Example 2. This strategy can however have a negative effect on the convergence behavior of the algorithm.

3.6 KLMS algorithm with forward-backward splitting

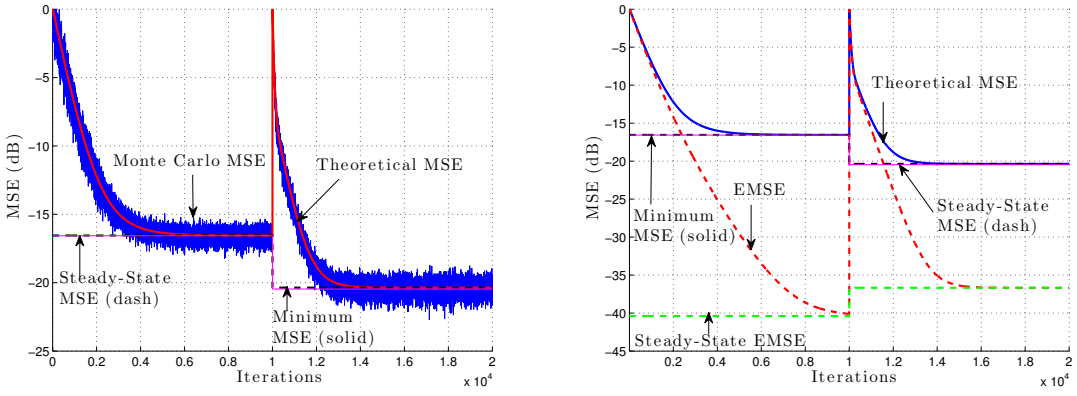
We shall now introduce a KLMS-type algorithm based on forward-backward splitting, which can automatically update the dictionary in an online way by discarding the obsolete and



(a) $\mathcal{D}_2 = \{15@\sqrt{0.0656}\}$

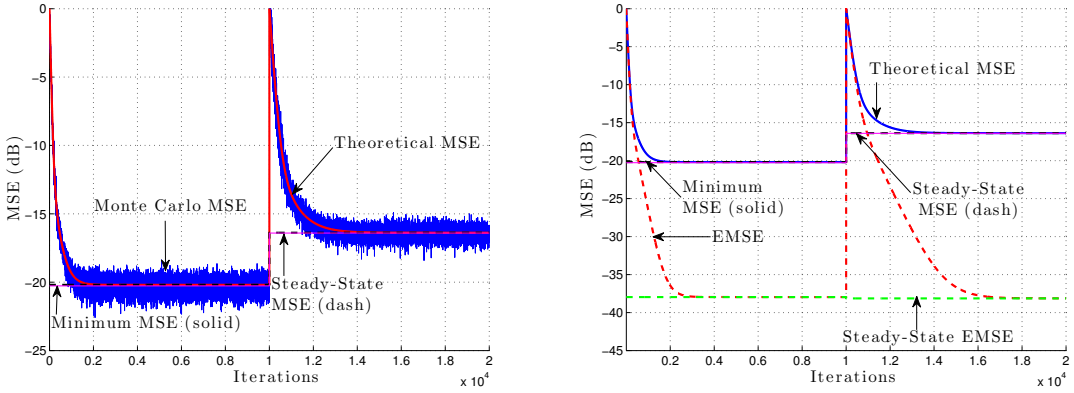


(b) $\mathcal{D}_2 = \{15@\sqrt{0.0156}\}$

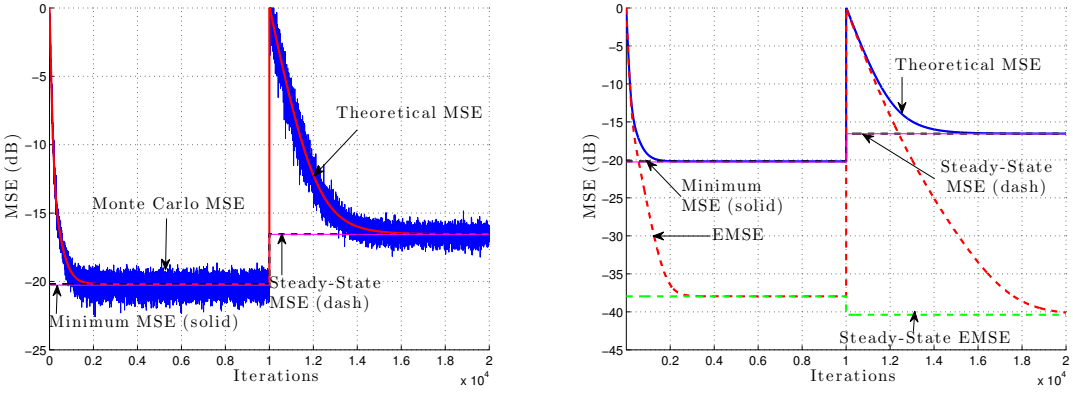


(c) $\mathcal{D}_2 = \{15@\sqrt{0.0156}\} \cup \{15@\sqrt{0.0656}\}$

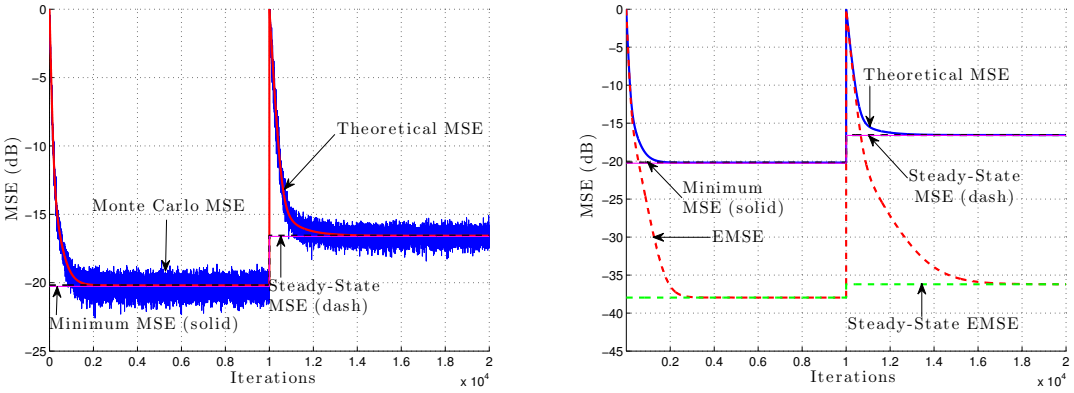
Figure 3.4: Learning curves for Example 2 with $\sigma_{u_2}, \sigma_{v_u} : \sqrt{0.0656} \to \sqrt{0.0156}$ and $\mathcal{D}_1 = \{15@\sqrt{0.0656}\}$. See the first row of Table 3.2.



(a) $\mathcal{D}_2 = \{15@\sqrt{0.0156}\}$



(b) $\mathcal{D}_2 = \{15@\sqrt{0.0656}\}$



(c) $\mathcal{D}_2 = \{15@\sqrt{0.0156}\} \cup \{15@\sqrt{0.0656}\}$

Figure 3.5: Learning curves for Example 2 with $\sigma_{u_2}, \sigma_{v_u} : \sqrt{0.0156} \to \sqrt{0.0656}$ and $\mathcal{D}_1 = \{15@\sqrt{0.0156}\}$. See the second row of Table 3.2.

invalid elements and adding appropriate ones.

3.6.1 Forward-backward splitting method in a nutshell

Consider first the following optimization problem [Bauschke 2011]

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}\in\mathbb{R}^N}{\arg\min} \left\{ Q(\boldsymbol{\alpha}) = J(\boldsymbol{\alpha}) + \lambda\Omega(\boldsymbol{\alpha}) \right\} \tag{3.42}$$

where $J(\cdot)$ is a convex empirical loss function with Lipschitz continuous gradient and Lipschitz constant $1/\eta_0$. Function $\Omega(\cdot)$ is a convex, continuous, but not necessarily differentiable regularizer, and λ is a positive regularization constant. This problem has been extensively studied in the literature, and can be solved with forward-backward splitting [Beck 2009]. In a nutshell, this approach consists of minimizing the following quadratic approximation of $Q(\boldsymbol{\alpha})$ at a given point $\boldsymbol{\alpha}_n$, in an iterative way,

$$Q_\eta(\boldsymbol{\alpha}, \boldsymbol{\alpha}_n) = J(\boldsymbol{\alpha}_n) + \nabla J(\boldsymbol{\alpha}_n)^\top (\boldsymbol{\alpha} - \boldsymbol{\alpha}_n) \; + \frac{1}{2\eta}\|\boldsymbol{\alpha} - \boldsymbol{\alpha}_n\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}) \tag{3.43}$$

since $Q(\boldsymbol{\alpha}) \leq Q_\eta(\boldsymbol{\alpha}, \boldsymbol{\alpha}_n)$ for any $\eta \leq \eta_0$. Simple algebra shows that the function $Q_\eta(\boldsymbol{\alpha}, \boldsymbol{\alpha}_n)$ admits a unique minimizer, denoted by $\boldsymbol{\alpha}_{n+1}$, given by

$$\boldsymbol{\alpha}_{n+1} = \underset{\boldsymbol{\alpha}\in\mathbb{R}^N}{\arg\min} \left\{ \lambda\Omega(\boldsymbol{\alpha}) + \frac{1}{2\eta}\|\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}_n\|_2^2 \right\} \tag{3.44}$$

with $\hat{\boldsymbol{\alpha}}_n = \boldsymbol{\alpha}_n - \eta\nabla J(\boldsymbol{\alpha}_n)$. It is interesting to note that $\hat{\boldsymbol{\alpha}}_n$ can be interpreted as an intermediate gradient descent step on the cost function $J(\cdot)$. Problem (3.44) is called the proximity operator for the regularizer $\Omega(\cdot)$, and is denoted by $\mathrm{Prox}_{\lambda\eta\Omega(\cdot)}(\cdot)$. While this method can be considered as a two-step optimization procedure, it is equivalent to a subgradient descent with the advantage of promoting exact sparsity at each iteration. The convergence of the optimization procedure (3.44) to a global minimum is ensured if $1/\eta$ is a Lipschitz constant of the gradient $\nabla J(\boldsymbol{\alpha})$. See [Bauschke 2011]. In the case $J(\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{d} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2$ considered in (2.58), where $\boldsymbol{K}$ is a $(N \times N)$ matrix, a well-established condition ensuring the convergence of $\boldsymbol{\alpha}_{n+1}$ to a minimizer of problem (3.42) is to require that [Beck 2009]

$$0 < \eta < 2/\mathrm{eig}_{\max}\{\boldsymbol{K}^\top\boldsymbol{K}\} \tag{3.45}$$

where $\mathrm{eig}_{\max}\{\cdot\}$ is the maximum eigenvalue. A companion bound will be derived hereafter for the stochastic gradient descent algorithm.

Forward-backward splitting is an efficient method for minimizing convex cost functions with sparse regularization. It was originally derived for offline learning but a generalization of this algorithm for stochastic optimization, the so-called FOBOS, was proposed in [Duchi 2009]. It consists of using a stochastic approximation for ∇J at each iteration. This online approach can be easily coupled with the KLMS algorithm but, for convenience of presentation, we shall now describe the offline setup based on problem (2.58) represented as follows:

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}\in\mathbb{R}^N}{\arg\min} \left\{ J(\boldsymbol{\alpha}) = \sum_{j=1}^{N} (d_j - \boldsymbol{\alpha}^\top\boldsymbol{\kappa}_j)^2 + \gamma\,\Omega(\boldsymbol{\alpha}) \right\}. \tag{3.46}$$

3.6.2 Application to KLMS algorithm

In order to automatically discard the irrelevant elements from the current dictionary $\mathcal{D}$, let us consider the minimization problem (2.58) with the sparsity-promoting convex regularization function $\Omega(\cdot)$

$$\boldsymbol{\alpha}^* = \underset{\boldsymbol{\alpha}\in\mathbb{R}^N}{\arg\min} \left\{ Q(\boldsymbol{\alpha}) = \|\boldsymbol{d} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}) \right\} \tag{3.47}$$

where $\boldsymbol{K}$ is the $(N \times N)$ Gram matrix with (i,j)-th entry $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$. Problem (3.47) is of the form (3.42), and can be solved with the forward-backward splitting method. Two regularization terms are considered.

Firstly, we suggest the use of the well-known ℓ_1-norm function defined as $\Omega_1(\boldsymbol{\alpha}) = \sum_m |\alpha(m)|$. This regularization function is often used for sparse regression and its proximity operator is separable. Its m-th entry can be expressed as [Bauschke 2011]

$$\left(\mathrm{Prox}_{\lambda\eta\|\cdot\|_1}(\boldsymbol{\alpha})\right)(m) = \mathrm{sign}\big\{\alpha(m)\big\} \max\big\{|\alpha(m)| - \lambda\eta,\, 0\big\}. \tag{3.48}$$

It is called the soft thresholding operator. One major drawback is that it promotes biased prediction.

Secondly, we consider an adaptive ℓ_1-norm function of the form $\Omega_a(\boldsymbol{\alpha}) = \sum_m w_m|\alpha(m)|$ where the $\{w_m\}_m$ is a set of weights to be dynamically adjusted. The proximity operator for this regularization function is defined by

$$\left(\mathrm{Prox}_{\lambda\eta\Omega_a(\cdot)}(\boldsymbol{\alpha})\right)(m) = \mathrm{sign}\big\{\alpha(m)\big\} \max\big\{|\alpha(m)| - \lambda\eta\, w_m,\, 0\big\}. \tag{3.49}$$

This regularization function has been proven to be more consistent than the usual ℓ_1-norm [Zou 2006], and tends to reduce the bias induced by the latter. Weights are usually chosen as $w_m = 1/(|\alpha_{\mathrm{opt}}(m)| + \epsilon_\alpha)$, where $\boldsymbol{\alpha}_{\mathrm{opt}}$ is the least-square solution of the problem (2.58), and ϵ_α a small constant to prevent the denominator from vanishing [Candès 2008]. Since $\boldsymbol{\alpha}_{\mathrm{opt}}$ is not available in our online case, we chose $w_m = 1/(|\alpha_{n-1}(m)| + \epsilon_\alpha)$ at each iteration n. This technique, also referred to as reweighted least-square, is performed at each iteration of the stochastic optimization process. Note that a similar regularization term was used in [Chen 2009] in order to approximate the ℓ_0-norm.

The pseudocode for KLMS algorithm with sparsity-promoting regularization, called FOBOS-KLMS, is provided in Algorithm 1. It can be noticed that the proximity operator is applied after the gradient descent step. The trivial dictionary elements associated with null coefficients in vector $\boldsymbol{\alpha}_n$ are eliminated. On the one hand, this approach reduces to the generic KLMS algorithm in the case $\lambda = 0$. On the other hand, FOBOS-KLMS appears to be the mono-kernel counterpart of the dictionary-refinement technique proposed in [Yukawa 2012] in the multi-kernel adaptive filtering context. The stability of this method is analyzed in the next subsection, which is an additional contribution of this chapter.

3.6.3 Stability in the mean

We shall now discuss the stability in the mean sense of the FOBOS-KLMS algorithm. We observe that the KLMS algorithm with the sparsity inducing regularization can be written

Algorithm 1 FOBOS-KLMS

Initialization
Select the step-size η, and the parameters of the kernel;
Insert $\kappa(\cdot, \boldsymbol{u}_1)$ into the dictionary, $\boldsymbol{\alpha}_1 = 0$.
for $n = 1, 2, \cdots$, do
  if $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| > \delta_0$
    Compute $\boldsymbol{\kappa}_{\omega,n}$ and $\boldsymbol{\alpha}_n$ using equation (3.2);
  elseif $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \delta_0$
    Incorporate $\kappa(\cdot, \boldsymbol{u}_n)$ into the dictionary;
    Compute $\boldsymbol{\kappa}_{\omega,n}$ and $\boldsymbol{\alpha}_n$ using equation (3.3);
  end if
  $\boldsymbol{\alpha}_n = \mathrm{Prox}_{\lambda\eta\Omega(\cdot)}(\boldsymbol{\alpha}_n)$ using (3.48) or (3.49);
  Remove $\kappa(\cdot, \boldsymbol{u}_{\omega_m})$ from the dictionary if $\alpha_n(m) = 0$.
  The solution is given as
    $\psi(\boldsymbol{u}_n) = \sum_{m=1}^{M} \alpha_m \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})$.
end for

as

$$\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n-1} + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n} - \boldsymbol{f}_{n-1} \tag{3.50}$$

with

$$f_{n-1}(m) = \begin{cases} \lambda\eta\, \mathrm{sign}\big(\hat{\boldsymbol{\alpha}}_{n-1}(m)\big) & \text{if} \quad |\hat{\boldsymbol{\alpha}}_{n-1}(m)| \geq \lambda\eta \\ \hat{\boldsymbol{\alpha}}_{n-1}(m) & \text{otherwise} \end{cases} \tag{3.51}$$

where $\hat{\boldsymbol{\alpha}}_n = \boldsymbol{\alpha}_{n-1} + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n}$. The function $\mathrm{sign}(\alpha)$ is defined by

$$\mathrm{sign}(\alpha) = \begin{cases} \alpha/|\alpha| & \alpha \neq 0; \\ 0 & \text{otherwise.} \end{cases} \tag{3.52}$$

Up to a variable change in λ, the general form (3.50)–(3.51) remains the same with the regularization function (3.49). Note that the sequence $|f_{n-1}(m)|$ is bounded, by $\lambda\eta$ for the operator (3.48), and by $\lambda\eta/\epsilon_\alpha$ for the operator (3.49).

Theorem 3.6.1 *Assume MIA holds. For any initial condition $\boldsymbol{\alpha}_0$, the KLMS algorithm with sparsity promoting regularization (3.48) and (3.49) asymptotically converge in the mean sense if the step-size η is chosen to satisfy*

$$0 < \eta < 2/\mathrm{eig}_{\max}\{\boldsymbol{R}_{\kappa\kappa}\} \tag{3.53}$$

where $\boldsymbol{R}_{\kappa\kappa} = E\{\boldsymbol{\kappa}_{\omega,n}\boldsymbol{\kappa}_{\omega,n}^\top\}$ is the $(M \times M)$ correlation matrix of the kernelized input $\boldsymbol{\kappa}_{\omega,n}$, and $\mathrm{eig}_{\max}\{\boldsymbol{R}_{\kappa\kappa}\}$ is the maximum eigenvalue of $\boldsymbol{R}_{\kappa\kappa}$.

Proof. To prove this theorem, we observe that the recursion (3.17) for the weight error vector $\boldsymbol{v}_n$ becomes

$$\boldsymbol{v}_n = \boldsymbol{v}_{n-1} - \eta\, \boldsymbol{\kappa}_{\omega,n}(\boldsymbol{\kappa}_{\omega,n}^\top\, \boldsymbol{v}_{n-1} + e_n^o) - \boldsymbol{f}_{n-1}. \tag{3.54}$$

Taking the expected value of both sides, and using the same assumptions as for (3.21), leads to

$$E\{\boldsymbol{v}_n\} = (\boldsymbol{I} - \eta\,\boldsymbol{R}_{\kappa\kappa})^n E\{\boldsymbol{v}_0\} + \sum_{i=0}^{n-1} (\boldsymbol{I} - \eta\,\boldsymbol{R}_{\kappa\kappa})^i E\{\boldsymbol{f}_{n-i-1}\} \tag{3.55}$$

with $\boldsymbol{v}_0$ the initial condition. To prove the convergence of $E\{\boldsymbol{v}_n\}$, we have to show that both terms on the r.h.s. converge as n goes to infinity. The first term converges to zero if we can ensure that $\nu \triangleq \|\boldsymbol{I} - \eta\,\boldsymbol{R}_{\kappa\kappa}\|_2 < 1$, where $\|\cdot\|_2$ denotes the 2-norm (spectral norm). We can easily check that this condition is met for any step-size η satisfying the condition (3.53) since

$$\nu = \max_{m=1,\ldots,M} \left|1 - \eta\,\mathrm{eig}_m\{\boldsymbol{R}_{\kappa\kappa}\}\right| \tag{3.56}$$

where $\mathrm{eig}_m\{\boldsymbol{R}_{\kappa\kappa}\}$ is the m-th eigenvalue of $\boldsymbol{R}_{\kappa\kappa}$. Let us show now that condition (3.53) also implies that the second term on the r.h.s. of equation (3.55) asymptotically converges to a finite value, thus leading to the overall convergence of this recursion. First it has been noticed that the sequence $|f_{n-1}(m)|$ is bounded. Thus, each term of this series is bounded because

$$\begin{aligned} \|(\boldsymbol{I} - \eta\,\boldsymbol{R}_{\kappa\kappa})^i E\{\boldsymbol{f}_{n-i-1}\}\|_2 &\leq \|(\boldsymbol{I} - \eta\,\boldsymbol{R}_{\kappa\kappa})^i\|_2 E\big\{\|\boldsymbol{f}_{n-i-1}\|_2\big\} \\ &\leq \sqrt{M}\,\nu^i\,f_{\max} \end{aligned} \tag{3.57}$$

where $f_{\max} = \lambda\eta$ or $\lambda\eta/\epsilon_\alpha$, depending if one uses the regularization function (3.48) or (3.49). Condition (3.53) implies that $\nu < 1$ and, as a consequence,

$$\sum_{i=0}^{n-1} \|\big(\boldsymbol{I} - \eta\,\boldsymbol{R}_{\kappa\kappa}\big)^i\; E\big\{\boldsymbol{f}_{n-i-1}\big\}\|_2 \leq \frac{\sqrt{M}\,f_{\max}}{1-\nu}. \tag{3.58}$$

The second term on the r.h.s. of equation (3.55) is an absolutely convergent series. This implies that it is a convergent series. □

Because the two terms of equation (3.55) are convergent series, we finally conclude that $E\{\boldsymbol{v}_n\}$ converges to a steady-state value if condition (3.53) is satisfied. Before concluding this section, it should be noticed that it was shown in [Parreira 2012] that

$$\mathrm{eig}_{\max}\{\boldsymbol{R}_{\kappa\kappa}\} = r_{\mathrm{md}} + (M-1)\,r_{\mathrm{od}}. \tag{3.59}$$

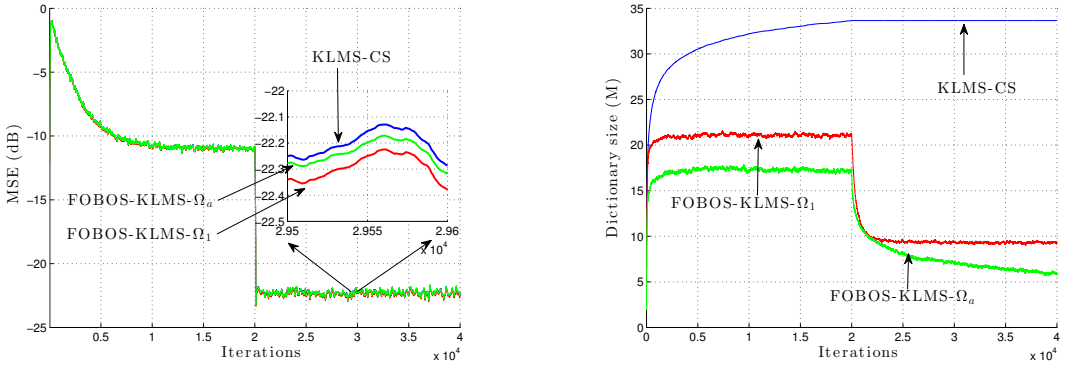
Parameters r_{md} and r_{od} are given by expression (3.13) in the case of a possibly partially matching dictionary.

3.7 Simulation results of proposed algorithm

We shall now illustrate the good performance of the FOBOS-KLMS algorithm with the two examples considered in this subection. Experimental settings were unchanged, and the results were averaged over 200 Monte Carlo runs. The coherence threshold δ_0 in Algorithm 1 was set to 0.01.

One can observe in Figures 3.7 and 3.9 that the size of the dictionary designed by the KLMS with coherence criterion dramatically increases when the variance of the input

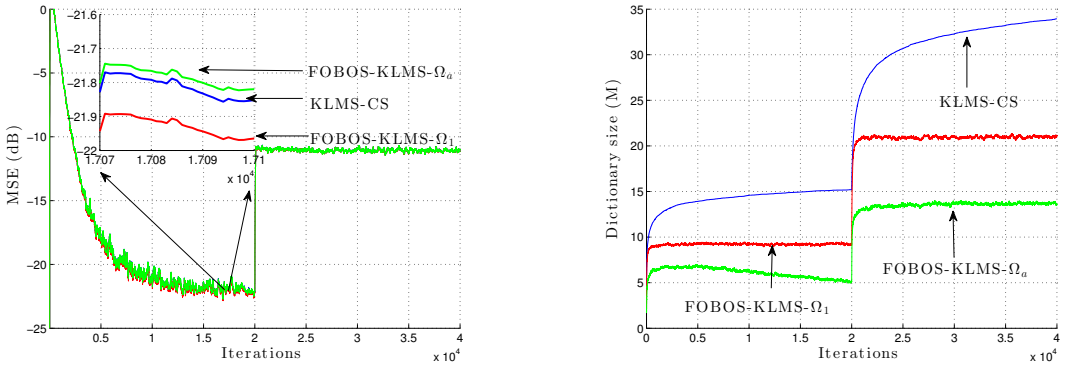
signal increases. In this case, this increased dynamic forces the algorithm to pave the input space $\mathcal{U}$ with additional dictionary elements. In Figures 3.6 and 3.8, the algorithm does not face this problem since the variance of the input signal abruptly decreases. The dictionary update with new elements is suddenly stopped. Again, these two scenarios clearly show the need for dynamically updating the dictionary by adding or discarding elements. Figures 3.6 to 3.9 clearly illustrate the merits of the FOBOS-KLMS algorithm with the regularizations (3.48) and (3.49). Both principles efficiently control the structure of the dictionary as a function of instantaneous characteristics of the input signal. They significantly reduce the order of the KLMS filter without affecting their performances.



(a) MSE

(b) Evolution of the size of dictionary

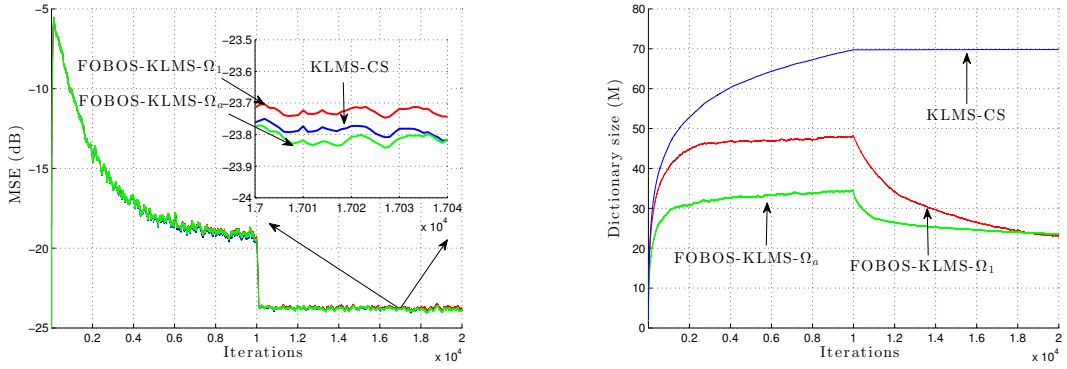
Figure 3.6: Learning curves for Example 1 where $\sigma_u : 0.35 \rightarrow 0.15$.



(a) MSE

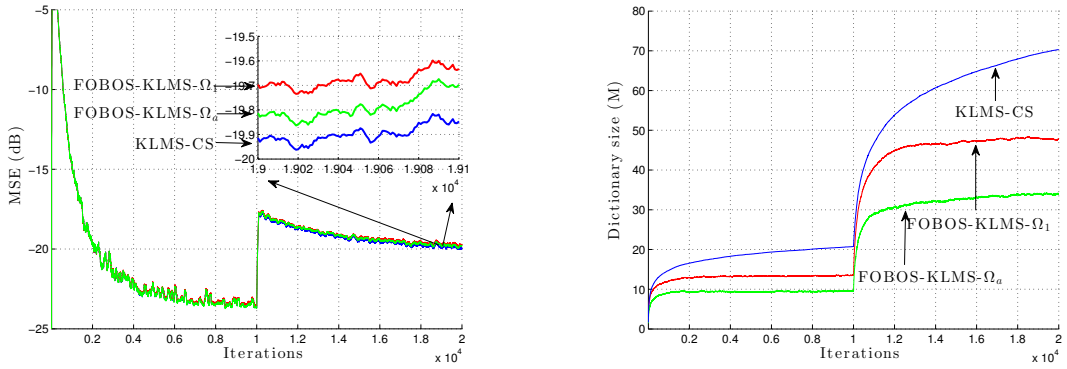
(b) Evolution of the size of dictionary

Figure 3.7: Learning curves for Example 1 where $\sigma_u : 0.15 \rightarrow 0.35$.



(a) MSE (b) Evolution of the size of dictionary

Figure 3.8: Learning curves for Example 2 with $\sigma_{u_2}, \sigma_{v_u}$: $\sqrt{0.0656} \rightarrow \sqrt{0.0156}$.



(a) MSE (b) Evolution of the size of dictionary

Figure 3.9: Learning curves for Example 2 with $\sigma_{u_2}, \sigma_{v_u}$: $\sqrt{0.0156} \rightarrow \sqrt{0.0656}$.

3.8 Conclusion

In this chapter, we presented an analytical study of the convergence behavior of the Gaussian KLMS algorithm in the case where the statistics of the dictionary elements only partially match the statistics of the input data. This allowed us to theoretically emphasize the need for updating the dictionary in an online way, by discarding the obsolete elements and adding appropriate ones. With the theoretical justification we then proposed the so-called FOBOS-KLMS algorithm, based on forward-backward splitting to deal with ℓ_1-norm regularization, in order to automatically adapt the dictionary to the instantaneous statistical characteristics of the input signal. Finally, the stability in the mean of this method was discussed, and a condition on the step-size for convergence was derived. The merits of FOBOS-KLMS were illustrated by simulation examples.

Chapter 4

Multikernel adaptive filtering algorithm

Contents

4.1 Introduction

During the last decade, multikernel learning has been extensively studied in the literature for classification and regression [Bach 2004, Sonnenburg 2006, Rakotomamonjy 2008, Kloft 2009, Martins 2011, Gonen 2011]. This strategy aims at learning simultaneously a predictor and its kernel in supervised learning settings. Multikernel learning methods have been shown to result in improved performance over single-kernel learning methods since they offer more flexibility to handle nonlinearities. Investigations on multikernel learning have focused almost exclusively on batch processing, and only few efforts have been directed toward online processing. It is only recently that this framework has been applied to kernel adaptive filtering. Several strategies were proposed in the literature, mostly of KLMS type. In a first category are the methods based on a convex combination of kernels with fixed coefficients. The kernelized data are then fed into a LMS filter. Flexibility is

provided by the kernel definition, which nevertheless does not vary over time. As it is no more than a simple monokernel LMS algorithm, we shall refer to it as the single-input multikernel LMS (SI-MKLMS) algorithm for short in the sequel. As practitioners may be interested in more flexible models in some situations, in a second category are the KLMS methods based on a linear combination of kernels that adapt over time. We shall refer to the approaches as multi-input multikernel LMS (MI-MKLMS) algorithms, which were introduced independently in [Yukawa 2012, Yukawa 2013] and [Tobar 2012c, Tobar 2014a]. These works also propose acceptance/rejection stages for kernel-based model selection in an online way. In [Tobar 2012c, Tobar 2014a], model selection is performed with novelty criterion.

Surprisingly, no theoretical analysis of MKLMS algorithm has already been proposed until now in the literature. The aim of this chapter is to provide such results when these algorithms operate with a linear combination of Gaussian kernels with different bandwidth parameters. This chapter is organized as follows. First, two distinct types of MKLMS algorithms are introduced in brief. Next, we present a theoretical analysis of the transient and steady-state behavior of the MI-MKLMS algorithm, in the mean and mean-square sense. Then we evaluate the accuracy of these analytical models by numerical simulations, and we compare with the performances of SI-MKLMS and MI-MKLMS. Finally, we provide some concluding remarks.

4.2 Multikernel LMS algorithms

In this section, we shall introduce two possible principles underlying the so-called multikernel LMS algorithms.

4.2.1 Single-input multikernel LMS algorithm

In some scenarios, practitioners may be interested in more flexible models than the monokernel adaptive filtering model (2.59). A convenient approach is then to consider that the kernel κ is actually a convex combination of kernels $\sum_{k=1}^{K} \beta_k \, \kappa_k$, that is, with $\beta_k \geq 0$ and $\sum_{k=1}^{K} \beta_k = 1$. The kernels κ_k can simply be usual kernels, such as Gaussian kernels, with different parameter settings. This is the rationale of the so-called SI-MKLMS presented in Figure 4.1, which is indeed a standard monokernel LMS that combines metric features from distinct RKHS for possibly more accurate or robust results. This should not be confused with the multiple kernel learning problem introduced in next subsection. This problem was introduced in [Lanckriet 2004] for classification, and widely studied since this pioneering work. With the SI-MKLMS algorithm, coefficients β_k are assumed to be fixed beforehand.

Let us start to describe the functional framework adopted for SI-MKLMS. Let $\{\kappa_k\}_{k=1}^{K}$ be the family of candidate kernels $\kappa_k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, and let $\mathcal{H}_k$ be the RKHS with sub-kernel κ_k. Then, the sum of sub-kernels $\kappa = \sum_{k=1}^{K} \beta_k \, \kappa_k$ with $\beta_k \geq 0$ is also a reproducing kernel. The corresponding RKHS, denoted by $\mathcal{H}_{\mathrm{SI}}$, contains the functions $\varphi = \sum_{k=1}^{K} \beta_k \, \varphi_k$ with

$\varphi_k \in \mathcal{H}_k$. The norm in $\mathcal{H}_{\text{SI}}$ is defined as [Aronszajn 1950]

$$\|\varphi\|_{\mathcal{H}_{\text{SI}}} = \min_{\beta_k}\{|\beta_1|\,\|\varphi_1\|_{\mathcal{H}_1} + \ldots + |\beta_K|\,\|\varphi_K\|_{\mathcal{H}_K} \text{ for all } \varphi = \beta_1\varphi_1 + \ldots + \beta_K\varphi_K,\ \varphi_k \in \mathcal{H}_k\} \tag{4.1}$$

If $\mathcal{H}_{\text{SI}}$ is the direct sum of spaces $\mathcal{H}_k$, that is $\bigcap_{k=1}^{K} \mathcal{H}_k = \emptyset$, then norm (4.1) reduces to $\|\varphi\|_{\mathcal{H}_{\text{SI}}} = \sum_{k=1}^{K} |\beta_k|\,\|\varphi_k\|_{\mathcal{H}_k}$. The counterpart of the finite-order model (2.59) in $\mathcal{H}_{\text{SI}}$ is given by

$$\psi(\cdot) = \sum_{m=1}^{M}\sum_{k=1}^{K} \alpha_m\,\beta_k\,\kappa_k(\cdot, \boldsymbol{u}_{\omega_m}) \tag{4.2}$$

with $\psi_k(\cdot) = \sum_{m=1}^{M} \alpha_m\,\kappa_k(\cdot, \boldsymbol{u}_{\omega_m})$, $\boldsymbol{\omega} = \{\kappa(\cdot, \boldsymbol{u}_{\omega_m})\}_{m=1}^{M}$ the dictionary and $\{\alpha_m\}_{m=1}^{M}$ the filter coefficients to be estimated using monokernel LMS algorithm. We can write equation (4.2) at time instant n as

$$\begin{aligned}\psi(\boldsymbol{u}_n) &= \sum_{m=1}^{M}\sum_{k=1}^{K} \alpha_m(n)\,\beta_k\,\kappa_k(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m}) \\ &= \boldsymbol{\alpha}_n^{\top}\boldsymbol{\kappa}_{\omega,n}^{\text{SI}}\end{aligned} \tag{4.3}$$

with

$$\boldsymbol{\alpha}_n = [\alpha_1(n), \ldots, \alpha_M(n)]^{\top} \tag{4.4}$$

and

$$\boldsymbol{\kappa}_{\omega,n}^{\text{SI}} = \left[\sum_{k=1}^{K} \beta_k\,\kappa_k(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}), \ldots, \sum_{k=1}^{K} \beta_k\,\kappa_k(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_M})\right]^{\top}. \tag{4.5}$$

The kernelized input vector $\boldsymbol{\kappa}_{\omega,n}^{\text{SI}}$ contains the input vector $\boldsymbol{u}_n$ mapped into $\mathcal{H}_k$ by each kernel κ_k. The block diagram of SI-MKLMS algorithm is depicted in Figure 4.1.

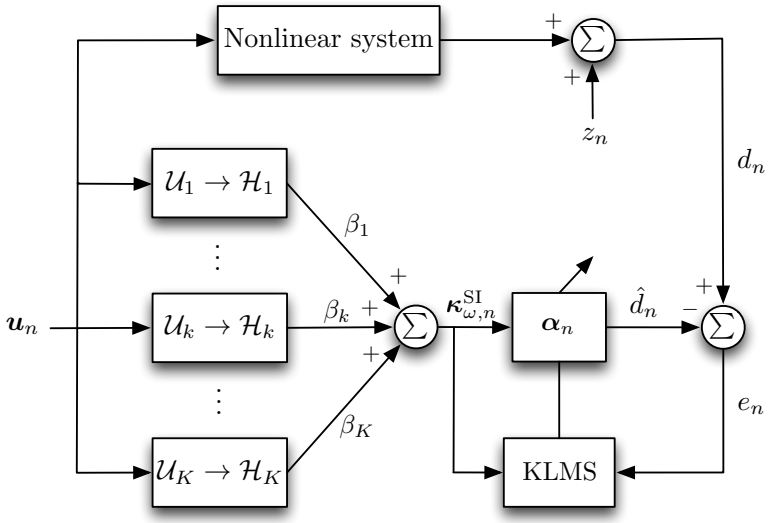


Figure 4.1: Single-input multikernel LMS algorithm.

From Figure 4.1, note that SI-MKLMS algorithm is a generalized standard monokernel LMS method, whose update rule is given by

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n}^{\mathrm{SI}}. \tag{4.6}$$

As monokernel LMS, the algorithm needs a Rejection/Acceptance stage at each time instant to update the dictionary. The CS criterion for single kernel can be extended to operate with multikernel algorithms as follows:

$$\max_{m=1,\ldots,M} |\bar{\kappa}(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \delta_0 \tag{4.7}$$

or

$$\max_{m=1,\ldots,M} \left| \frac{\sum_{k=1}^{K} \kappa_k(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})}{\sqrt{\sum_{k=1}^{K} \kappa_k(\boldsymbol{u}_n, \boldsymbol{u}_n)}\sqrt{\sum_{k=1}^{K} \kappa_k(\boldsymbol{u}_{\omega_m}, \boldsymbol{u}_{\omega_m})}} \right| \leq \delta_0. \tag{4.8}$$

4.2.2 Multi-input multikernel LMS algorithm

Another convenient approach is to linearly combine multiple monokernel models in some optimal least-squares sense, and to enable these models to adapt over time. This strategy was introduced in [Yukawa 2012, Yukawa 2013] and [Tobar 2012c, Tobar 2014a]. The authors in [Tobar 2014a] introduced an interesting functional framework based on the concept of vector-valued RKHS to support this strategy. Similarly, let $\{\kappa_k\}_{k=1}^{K}$ be the family of candidate kernels $\kappa_k : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$, and let $\mathcal{H}_k$ be the RKHS with kernel κ_k. Kernels κ_k can simply be usual kernels, such as Gaussian kernels, with different parameter settings. Consider the multidimensional mapping

$$\begin{aligned} \Phi : \quad & \mathcal{U} \longrightarrow \mathcal{M} \\ & \boldsymbol{u} \longmapsto \Phi(\boldsymbol{u}) = (\varphi_1(\boldsymbol{u}), \ldots, \varphi_K(\boldsymbol{u}))^{\top} \end{aligned} \tag{4.9}$$

with $\varphi_k \in \mathcal{H}_k$, and the inner product in $\mathcal{M}$ defined as follows: $\langle \Phi, \Phi' \rangle_{\mathcal{M}} = \sum_{k=1}^{K} \langle \varphi_k, \varphi_k' \rangle_{\mathcal{H}_k}$. The space $\mathcal{M}$ of vector-valued functions endowed with the inner product $\langle \cdot,\cdot \rangle_{\mathcal{M}}$ is a Hilbert space as $(\mathcal{H}_k, \langle \cdot,\cdot \rangle_{\mathcal{H}_k})$ is a Hilbert space for all k. We can define the vector-valued representer of evaluation

$$\begin{aligned} \Phi(\boldsymbol{u}) &= [\Phi(\cdot), \boldsymbol{\kappa}_{\mathcal{M}}(\cdot, \boldsymbol{u})] \\ &= \big(\langle \varphi_1, \kappa_1(\cdot, \boldsymbol{u}) \rangle_{\mathcal{H}_1}, \ldots, \langle \varphi_K, \kappa_K(\cdot, \boldsymbol{u}) \rangle_{\mathcal{H}_K}\big)^{\top} \end{aligned} \tag{4.10}$$

where $[\cdot,\cdot]$ denotes the entrywise inner product, and $\boldsymbol{\kappa}_{\mathcal{M}}(\cdot, \boldsymbol{u}) = \big(\kappa_1(\cdot, \boldsymbol{u}), \ldots, \kappa_K(\cdot, \boldsymbol{u})\big)^{\top}$. This yields the following reproducing property

$$\boldsymbol{\kappa}_{\mathcal{M}}(\boldsymbol{u}, \boldsymbol{u}') = [\boldsymbol{\kappa}_{\mathcal{M}}(\cdot, \boldsymbol{u}), \boldsymbol{\kappa}_{\mathcal{M}}(\cdot, \boldsymbol{u}')]. \tag{4.11}$$

Let $\Psi(\boldsymbol{u})$ be a function in $\mathcal{M}$, and let $\psi(\boldsymbol{u}) = \sum_{k=1}^{K} \psi_k(\boldsymbol{u})$ denote the scalar-valued function that sums the entries of the vector-valued function $\Psi(\boldsymbol{u})$, that is, $\psi(\boldsymbol{u}) = \mathbf{1}^{\top}\Psi(\boldsymbol{u})$ where $\mathbf{1}$

is a column vector of ones. The MKLMS algorithm aims at estimating a multidimensional function Ψ in $\mathcal{M}$ that minimizes the regularized least-square error

$$\begin{aligned}\min_{\Psi\in\mathcal{M}} J(\Psi) &= \frac{1}{K}\sum_{n=1}^{N}\|\boldsymbol{d}_n-\Psi(\boldsymbol{u}_n)\|_2^2+\mu\|\Psi\|_{\mathcal{M}}^2\\ &= \frac{1}{K}\sum_{n=1}^{N}\sum_{k=1}^{K}[d_n-\psi_k(\boldsymbol{u}_n)]^2+\mu\sum_{k=1}^{K}\|\psi_k\|_{\mathcal{H}_k}^2\end{aligned} \tag{4.12}$$

with μ a nonnegative regularization constant, and $\boldsymbol{d}_n = d_n\mathbf{1}$. Let us recall that the directional Fréchet derivative of J, in the direction $\gamma\in\mathcal{M}$ and at the point $\Psi\in\mathcal{M}$, is defined as

$$\partial_\gamma J(\Psi) = \lim_{\varepsilon\to 0}\frac{J(\Psi+\varepsilon\gamma)-J(\Psi)}{\varepsilon}. \tag{4.13}$$

The gradient $\nabla J(\Psi)$ of J at the point Ψ, if it exists, satisfies

$$\partial_\gamma J(\Psi) = \langle\nabla J(\Psi),\gamma\rangle_{\mathcal{M}} \tag{4.14}$$

for all $\gamma\in\mathcal{M}$. Consider the functions $T_1(\Psi)=\Psi(\boldsymbol{u})$, given $\boldsymbol{u}\in\mathcal{U}$, and $T_2(\Psi)=\|\Psi\|_{\mathcal{M}}$ from $\mathcal{M}$ to $\mathbb{R}$. Using definition (4.14), it can be shown that

$$\nabla T_1(\Psi) = \boldsymbol{\kappa}_{\mathcal{M}}(\cdot,\boldsymbol{u}) \qquad\qquad \nabla T_2(\Psi) = 2\Psi \tag{4.15}$$

Using these expressions to calculate the gradient of the cost function (4.12), and equating it to zero, yields

$$\Psi(\cdot) = \left[\sum_{n=1}^{N}\alpha_{n,1}\,\kappa_1(\cdot,\boldsymbol{u}_n),\ldots,\sum_{n=1}^{N}\alpha_{n,K}\,\kappa_K(\cdot,\boldsymbol{u}_n)\right]^\top \tag{4.16}$$

where each $\boldsymbol{\alpha}_k=[\alpha_{1,k},\ldots,\alpha_{N,k}]^\top$ is the unique solution of the linear system (4.17), that is, by substituting (4.16) into (4.12) and equating to 0 the derivatives of the resulting cost with respect to $\boldsymbol{\alpha}_k$

$$(\boldsymbol{K}_k+\mu\boldsymbol{I})\,\boldsymbol{\alpha}_k = \boldsymbol{d} \tag{4.17}$$

where $\boldsymbol{K}_k$ is the Gram matrix with (i,j)-th entry denoted by $\kappa_k(\boldsymbol{u}_i,\boldsymbol{u}_j)$, $\boldsymbol{I}$ is the identity matrix, and $\boldsymbol{d}=[d_1,\ldots,d_N]^\top$. Finally, the optimal function $\psi(\boldsymbol{u})=\sum_{k=1}^{K}\psi_k(\boldsymbol{u})$ is given by

$$\psi(\cdot) = \sum_{n=1}^{N}\sum_{k=1}^{K}\alpha_{n,k}\,\kappa_k(\cdot,\boldsymbol{u}_n). \tag{4.18}$$

Using the generalized CS criterion defined by (4.7) for controlling model order, the finite-order model considered with MI-MKLMS algorithm is given by

$$\psi(\cdot) = \sum_{m=1}^{M}\sum_{k=1}^{K}\alpha_{m,k}\,\kappa_k(\cdot,\boldsymbol{u}_{\omega_m}). \tag{4.19}$$

The estimate of MI-MKLMS algorithm at time instant n can be written as

$$\begin{aligned}\psi(\boldsymbol{u}_n) &= \sum_{m=1}^{M}\sum_{k=1}^{K} \alpha_{m,k}(n)\,\kappa_k(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m}) \\ &= \boldsymbol{\alpha}_n^{\top}\boldsymbol{\kappa}_{\omega,n}^{\mathrm{MI}}\end{aligned} \tag{4.20}$$

where the weight vector and the multi-kernelized input vector are denoted by

$$\boldsymbol{\alpha}_n = \mathrm{vec}\big\{\boldsymbol{\alpha}_1(n), \ldots, \boldsymbol{\alpha}_K(n)\big\} \tag{4.21}$$

$$\boldsymbol{\kappa}_{\omega,n}^{\mathrm{MI}} = \mathrm{vec}\big\{\boldsymbol{\kappa}_{\omega,n}^{(1)}, \ldots, \boldsymbol{\kappa}_{\omega,n}^{(K)}\big\} \tag{4.22}$$

with $\boldsymbol{\alpha}_k = [\alpha_{1,k}(n), \ldots, \alpha_{M,k}(n)]^{\top}$ and $\boldsymbol{\kappa}_{\omega,n}^{(k)} = [\kappa_k(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_1}), \ldots, \kappa_k(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_M})]^{\top}$. In these expressions, operator $\mathrm{vec}\{\cdot\}$ stacks all its arguments on top of each other. The structure of MI-MKLMS algorithm is shown in Figure 4.2. Therefore, the recursive update of weight vector is given by

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n}^{\mathrm{MI}}. \tag{4.23}$$

Lemma 4.2.1 *The Wiener solution does not allow for the estimation of MI-MKLMS algorithm to be factorized into being exactly equal to the sub-kernel solution of SI-MKLMS algorithm, when the sub-kernel are selected from the disjoint RKHS* $\{\mathcal{H}_k\}_{k=1}^{K}$*, that is*

$$\sum_{k=1}^{K}\sum_{m=1}^{M} \alpha_{m,k}\,\kappa_k(\cdot, \boldsymbol{u}_{\omega_m}) \neq \sum_{m=1}^{M} \alpha_m \left(\sum_{k=1}^{K} \beta_k\,\kappa_k(\cdot, \boldsymbol{u}_{\omega_m})\right). \tag{4.24}$$

The proof is referred to [Tobar 2012c]. Obviously, Lemma 4.2.1 presents another significant discrimination between the two so-called *multi-kernel* algorithms from the perspective of mathematical equivalence of estimation functional.

The SI-MKLMS algorithm can be regarded as a sub-case for MI-MKLMS, which considers a more general setting. In addition, the convergence analysis of the SI-MKLMS can be resorted to our analysis of single Gaussian KLMS with fixed-dictionary derived in [Chen 2014a]. Therefore, in this thesis, the analysis of the convergence behavior of multikernel LMS will only be considered in the case of MI-MKLMS in the following. The convergence behavior of SI-MKLMS will however be compared with MI-MKLMS one in the experiment section. It needs to be pointed out that the multiple sub-kernel functions in the analysis are assumed to be Gaussian kernel functions with distinct bandwidths.

4.2.3 Optimal solution

Consider the mean-square error criterion

$$E\{e_n^2\} = \int_{\Omega}\int_{\mathcal{S}} e_n^2\, d\rho_{\mathcal{S}}(\boldsymbol{u}, \boldsymbol{d}\,|\,\boldsymbol{\omega})\, d\rho_{\Omega} \tag{4.25}$$

where $\rho_{\mathcal{S}}$ and ρ_{Ω} are Borel probability measures on $\mathcal{S} = \mathcal{U} \times \mathbb{R}$ and on the dictionary space Ω, respectively. Except with assumptions such as in [Parreira 2012], where the authors consider that the dictionary elements are governed by the same probability density function as

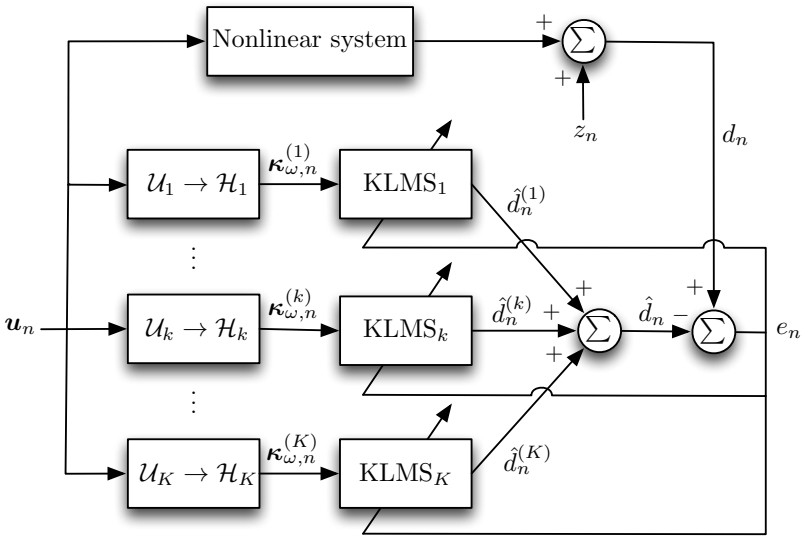


Figure 4.2: Multiple-input multikernel LMS algorithm.

the input data, the distribution of dictionary elements $\boldsymbol{\omega}$ generated by a dictionary learning method cannot usually be expressed in closed form. In this chapter, as introduced for the first time in [Chen 2014a], we consider the dictionary as a part of the filter parameters to be set. This pave the way for future works on dictionary design techniques driven by filter performance. Our objective here is to characterize both transient and steady-state of the mean-square error conditioned on $\boldsymbol{\omega}$, that is,

$$E_{\mathcal{S}}\{e_n^2 \,|\, \boldsymbol{\omega}\} = \int_{\mathcal{S}} e_n^2 \, d\rho_{\mathcal{S}}(\boldsymbol{u}, \boldsymbol{d} \,|\, \boldsymbol{\omega}). \tag{4.26}$$

We shall use the subscript ω for quantities conditioned on the dictionary, and $\mathcal{S}$ for expectations with respect to input data distribution.

We shall now derive the optimal solution of the MSE problem with the multikernel model (4.20). Given the dictionary $\boldsymbol{\omega} = \{\kappa_1(\cdot, \boldsymbol{u}_{\omega_m}), \ldots, \kappa_K(\cdot, \boldsymbol{u}_{\omega_m})\}_{m=1}^{M}$ and the multikernelized input vector $\boldsymbol{\kappa}_{\omega,n}$ defined by (4.22), the estimated system output shown in Figure 4.2 is defined as

$$\hat{d}_n = \boldsymbol{\alpha}^\top \boldsymbol{\kappa}_{\omega,n}. \tag{4.27}$$

The estimation error at time instant n is given by

$$e_{\omega,n} = d_n - \hat{d}_n. \tag{4.28}$$

Squaring both sides of this equation, and taking the expected value, leads to the MSE criterion

$$J_{\text{MSE}}(\boldsymbol{\alpha}) = E\{e_{\omega,n}^2\} = E\{d_n^2\} - 2\,\boldsymbol{p}_{\kappa d}^\top \boldsymbol{\alpha} + \boldsymbol{\alpha}^\top \boldsymbol{R}_{\kappa\kappa}\, \boldsymbol{\alpha} \tag{4.29}$$

where

$$\begin{aligned} \boldsymbol{R}_{\kappa\kappa} &= E\{\boldsymbol{\kappa}_{\omega,n}\, \boldsymbol{\kappa}_{\omega,n}^\top | \boldsymbol{\omega}\} \\ &= E\{\text{vec}\{\boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_K\}\, \text{vec}\{\boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_K\}^\top | \boldsymbol{\omega}\} \end{aligned} \tag{4.30}$$

is the correlation matrix of the multi-kernelized input given by $\boldsymbol{\omega}$, and

$$\boldsymbol{p}_{\kappa d} = E\{d_n\, \boldsymbol{\kappa}_{\omega,n} | \boldsymbol{\omega}\} \tag{4.31}$$

is the cross-correlation vector between $\boldsymbol{\kappa}_{\omega,n}$ and d_n. Assuming that $\boldsymbol{R}_{\kappa\kappa}$ is positive definite, the optimum weight vector is given by

$$\boldsymbol{\alpha}_{\text{opt},\omega} = \arg\min_{\boldsymbol{\alpha}} J_{\text{MSE}}(\boldsymbol{\alpha}) = \boldsymbol{R}_{\kappa\kappa}^{-1}\, \boldsymbol{p}_{\kappa d} \tag{4.32}$$

and the minimum MSE is

$$J_{\min} = E\{d_n^2\} - \boldsymbol{p}_{\kappa d}^{\top}\, \boldsymbol{R}_{\kappa\kappa}^{-1}\, \boldsymbol{p}_{\kappa d}. \tag{4.33}$$

These are the well-known expressions of the Wiener solution and minimum MSE, where the usual input signal vector is replaced by the multi-kernelized input.

4.3 Convergence behavior analysis of MI-MKLMS algorithm

4.3.1 Preliminaries and assumptions

We shall now derive the convergence model and stability conditions for the MI-MKLMS algorithm, given $\boldsymbol{\omega}$. Input data $\boldsymbol{u}_n$ are assumed to be zero-mean mean Gaussian random vectors with autocorrelation matrix $\boldsymbol{R}_{\boldsymbol{uu}} = E\{\boldsymbol{u}_n \boldsymbol{u}_n^{\top}\}$. Let $\boldsymbol{v}_{\omega,n}$ be the weight-error vector defined as

$$\boldsymbol{v}_{\omega,n} = \boldsymbol{\alpha}_n - \boldsymbol{\alpha}_{\text{opt},\omega}. \tag{4.34}$$

Before starting to derive the model, let us recall the following result on the moment generating function of any quadratic form of a Gaussian vector. Let $\boldsymbol{y} = [y_1, \ldots, y_q]^{\top}$ be a random vector following Gaussian distribution with zero-mean and covariance matrix

$$E\{\boldsymbol{y}\} = 0 \quad \text{and} \quad \boldsymbol{R}_{\boldsymbol{yy}} = E\{\boldsymbol{y}\, \boldsymbol{y}^{\top}\}. \tag{4.35}$$

Let the random variable τ be the quadratic form of $\boldsymbol{y}$ defined as

$$\tau = \boldsymbol{y}^{\top} \boldsymbol{H} \boldsymbol{y} + \boldsymbol{b}^{\top} \boldsymbol{y} \tag{4.36}$$

with vector $\boldsymbol{b} \in \mathbb{R}^q$. The moment generating function of τ is given by [Omura 1965]

$$\Psi_{\tau}(s) = |\boldsymbol{I} - 2s\, \boldsymbol{H}\, \boldsymbol{R}_{\boldsymbol{yy}}|^{-\frac{1}{2}} \cdot \exp\left(\frac{s^2}{2} \boldsymbol{b}^{\top} \boldsymbol{R}_{\boldsymbol{yy}} (\boldsymbol{I} - 2s\, \boldsymbol{H}\, \boldsymbol{R}_{\boldsymbol{yy}})^{-1} \boldsymbol{b} \right). \tag{4.37}$$

This result will be very useful to determine the expected values in the sequel. Simplifying assumptions are required in order to make the study of the stochastic behavior of $\boldsymbol{v}_{\omega,n}$ mathematically trackable. The following statistical assumption is required in the analysis:

Assumption 1 *$\boldsymbol{\kappa}_{\omega,n}\, \boldsymbol{\kappa}_{\omega,n}^{\top}$ is independent of $\boldsymbol{v}_{\omega,n}$.*

This assumption, called modified independence assumption (MIA), is justified in details in [Minkoff 2001]. It was successfully employed in several adaptive filter analyses, and was shown in [Minkoff 2001] to be less restrictive than the independence assumption (IA) [Sayed 2003]. It is named here for further reference *conditioned* MIA (CMIA) to distinguish from the MIA used in [Parreira 2012], since these variables are conditioned on $\boldsymbol{\omega}$.

4.3.2 Mean weight error analysis

Substituting (4.34) and (4.27) into (4.28), the estimation error $e_{\omega,n}$ can be expressed as

$$e_{\omega,n} = d_n - \boldsymbol{\kappa}_{\omega,n}^{\top}\, \boldsymbol{v}_{\omega,n} - \boldsymbol{\kappa}_{\omega,n}^{\top}\, \boldsymbol{\alpha}_{\mathrm{opt},\omega} \tag{4.38}$$

and the optimal estimation error is

$$e_{\mathrm{opt},\omega} = d_n - \boldsymbol{\kappa}_{\omega,n}^{\top}\, \boldsymbol{\alpha}_{\mathrm{opt},\omega}. \tag{4.39}$$

Then replacing (4.38) into (4.23) and using the definition of $\boldsymbol{v}_{\omega,n}$, we obtain the following the weight-error vector recursive equation

$$\boldsymbol{v}_{\omega,n+1} = \boldsymbol{v}_{\omega,n} + \eta\, d_n\, \boldsymbol{\kappa}_{\omega,n} - \eta\, \boldsymbol{\kappa}_{\omega,n}^{\top}\, \boldsymbol{v}_{\omega,n}\, \boldsymbol{\kappa}_{\omega,n} - \eta\, \boldsymbol{\kappa}_{\omega,n}^{\top}\, \boldsymbol{\alpha}_{\mathrm{opt},\omega}\, \boldsymbol{\kappa}_{\omega,n}. \tag{4.40}$$

Taking expected values of both sides of (4.40), using CMIA and (4.32), lead to the mean weight error model

$$E\{\boldsymbol{v}_{\omega,n+1}\} = (\boldsymbol{I} - \eta\, \boldsymbol{R}_{\kappa\kappa})\, E\{\boldsymbol{v}_{\omega,n}\}. \tag{4.41}$$

Let us evaluate the correlation matrix of kernelized inputs of MI-MKLMS. We introduce first some definitions:

$$\begin{aligned} \hat{i} &= \bar{i} + (i-1)M; \\ \hat{j} &= \bar{j} + (j-1)M; \end{aligned} \tag{4.42}$$

with $1 \le i,\, j \le K$ and $1 \le \bar{i},\, \bar{j} \le M$, then we have $1 \le \hat{i},\, \hat{j} \le KM$. Moreover, let us define

$$\xi_q = \begin{cases} \xi_1, & 1 \le \bar{q} + (q-1)M \le M; \\ & \vdots \\ \xi_K, & 1 + (K-1)M \le \bar{q} + (q-1)M \le KM; \end{cases} \tag{4.43}$$

with $1 \le \bar{q} \le M$ and $1 \le q \le K$. Consequently, the kernel bandwidths ξ_i and ξ_j used in the analysis can be determined by (4.43). The entries of the correlation matrix $\boldsymbol{R}_{\kappa\kappa}$ are given by

$$\begin{aligned} [\boldsymbol{R}_{\kappa\kappa}]_{\hat{i},\hat{j}} &= [\boldsymbol{R}_{\kappa\kappa}]_{\bar{i}+(i-1)M,\,\bar{j}+(j-1)M} \\ &= E_{\boldsymbol{u}} \left\{ \exp\left(-\frac{\xi_j^2 \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_{\hat{i}}}\|^2 + \xi_i^2 \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_{\hat{j}}}\|^2}{2\varepsilon^2} \right) \right\} \\ &= \exp\left(-\frac{1}{2\varepsilon^2} [\xi_j^2 \|\boldsymbol{u}_{\omega_{\hat{i}}}\|^2 + \xi_i^2 \|\boldsymbol{u}_{\omega_{\hat{j}}}\|^2] \right) \cdot E_{\boldsymbol{u}} \left\{ \exp\left(-\frac{1}{2\varepsilon^2} [\epsilon^2 \|\boldsymbol{u}_n\|^2 - 2(\xi_j^2 \boldsymbol{u}_{\omega_{\hat{i}}} + \xi_i^2 \boldsymbol{u}_{\omega_{\hat{j}}})^{\top} \boldsymbol{u}_n] \right) \right\} \end{aligned} \tag{4.44}$$

with $\varepsilon = \xi_i\, \xi_j$ and $\epsilon = (\xi_i^2 + \xi_j^2)^{\frac{1}{2}}$. Let us introduce the notations

$$\begin{aligned} \hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}}} &= \xi_j^2\, \boldsymbol{u}_{\omega_{\hat{i}}} + \xi_i^2\, \boldsymbol{u}_{\omega_{\hat{j}}}; \\ \|\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}}}\|^{(2)} &= \xi_j^2\, \|\boldsymbol{u}_{\omega_{\hat{i}}}\|^2 + \xi_i^2\, \|\boldsymbol{u}_{\omega_{\hat{j}}}\|^2. \end{aligned} \tag{4.45}$$

Comparing the second term on the RHS of (4.44) with (4.36), we can obtain $\boldsymbol{H} = \epsilon^2 \boldsymbol{I}$, $\boldsymbol{b} = -2(\xi_j^2 \boldsymbol{u}_{\omega_{\bar{i}}} + \xi_i^2 \boldsymbol{u}_{\omega_{\bar{j}}})$ and $s = -1/2\varepsilon^2$. Then we get

$$\begin{aligned}[\boldsymbol{R}_{\kappa\kappa}]_{\hat{i},\hat{j}} &= [\boldsymbol{R}_{\kappa\kappa}]_{\bar{i}+(i-1)M,\,\bar{j}+(j-1)M} \\ &= \exp\left(-\frac{1}{2\varepsilon^2}\|\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}}}\|^{(2)}\right)|\boldsymbol{I} + (\epsilon/\varepsilon)^2 \boldsymbol{R}_{\boldsymbol{uu}}|^{-\frac{1}{2}} \cdot \exp\left(-\frac{1}{2\varepsilon^4}\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}}} \boldsymbol{R}_{\boldsymbol{uu}}(\boldsymbol{I} + (\varepsilon/\epsilon)^2 \boldsymbol{R}_{\boldsymbol{uu}}^{-1})^{-1} \hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}}}\right).\end{aligned} \tag{4.46}$$

In order to express this formula in a more compact manner, we use the identity $(\boldsymbol{I} + \boldsymbol{A}^{-1})^{-1} = \boldsymbol{A}(\boldsymbol{A} + \boldsymbol{I})^{-1}$ with $\boldsymbol{R}_{\boldsymbol{uu}}(\boldsymbol{I} + (\epsilon/\varepsilon)^2 \boldsymbol{R}_{\boldsymbol{uu}})^{-1}$. Equation (4.46) can be rewritten as

$$\begin{aligned}[\boldsymbol{R}_{\kappa\kappa}]_{\hat{i},\hat{j}} &= [\boldsymbol{R}_{\kappa\kappa}]_{\bar{i}+(i-1)M,\,\bar{j}+(j-1)M} \\ &= |\boldsymbol{I} + (\epsilon/\varepsilon)^2 \boldsymbol{R}_{\boldsymbol{uu}}|^{-\frac{1}{2}} \cdot \exp\left(-\frac{1}{2\varepsilon^2\epsilon^2}\left[\epsilon^2\|\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}}}\|^{(2)} - \|\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}}}\|^2_{(\boldsymbol{I}+(\varepsilon/\epsilon)^2 \boldsymbol{R}_{\boldsymbol{uu}}^{-1})^{-1}}\right]\right).\end{aligned} \tag{4.47}$$

Theorem 4.3.1 *(Stability in the mean) Assume CMIA holds. Then, for any initial condition, given a dictionary $\boldsymbol{\omega}$, the MI-MKLMS algorithm with Gaussian kernels (4.23) asymptotically converges in the mean if the step-size is chosen to satisfy*

$$0 < \eta < \frac{2}{\mathrm{eig}_{\max}(\boldsymbol{R}_{\kappa\kappa})} \tag{4.48}$$

where $\mathrm{eig}_{\max}(\cdot)$ denotes the maximum eigenvalue of the matrix. The entries of $\boldsymbol{R}_{\kappa\kappa}$ are given by (4.44).

Proof. Using the unitary similarity transformation, the correlation matrix of multikernelized input vector $\boldsymbol{R}_{\kappa\kappa}$ can be expressed as

$$\boldsymbol{R}_{\kappa\kappa} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^\top \tag{4.49}$$

where $\boldsymbol{\Lambda} = \mathrm{Diag}(\lambda_1, \cdots, \lambda_M)$ is a diagonal matrix comprising all positive and real eigenvalues of the correlation matrix $\boldsymbol{R}_{\kappa\kappa}$, and $\boldsymbol{Q}$ is the matrix of the corresponding eigenvectors associated with the eigenvalues. Premultiplying both sides of (4.41) by $\boldsymbol{Q}^\top$ and using the matrix property $\boldsymbol{Q}^\top = \boldsymbol{Q}^{-1}$, yields

$$\boldsymbol{Q}^\top E\{\boldsymbol{v}_{\omega,n+1}\} = (\boldsymbol{I} - \eta\,\boldsymbol{\Lambda})\,\boldsymbol{Q}^\top E\{\boldsymbol{v}_{\omega,n}\}. \tag{4.50}$$

We define a new set of coordinates $\boldsymbol{v}'_{\omega,n+1} = \boldsymbol{Q}^\top E\{\boldsymbol{v}_{\omega,n+1}\}$, which represents the projection of the vector $E\{\boldsymbol{v}_{\omega,n+1}\}$ onto the eigenvectors of $\boldsymbol{R}_{\kappa\kappa}$. Then equation (4.41) can be rewritten as

$$\boldsymbol{v}'_{\omega,n+1} = (\boldsymbol{I} - \eta\,\boldsymbol{\Lambda})\,\boldsymbol{v}'_{\omega,n}. \tag{4.51}$$

We thus have $v'_{\omega,n+1}(m) = (1 - \eta\lambda_m)v'_{\omega,n}(m)$, $m = 1, \ldots, KM$. As $v'_{\omega,n+1}(m) = (1 - \eta\lambda_m)^n v'_{\omega,0}(m)$, (4.51) converges to zeros if $|1 - \eta\lambda_m| < 1$ for all m. This leads to the condition of stability in the mean (4.48). □

4.3.3 Mean squared error analysis

Using (4.38) and the CMIA, the second-order moments of the weights are related to the MSE through [Sayed 2003]

$$J_{\text{MSE}}(n) = J_{\min} + \text{trace}\{\boldsymbol{R}_{\kappa\kappa}\,\boldsymbol{C}_{v,\omega}(n)\} \tag{4.52}$$

where $\boldsymbol{C}_{v,\omega}(n) = E\{\boldsymbol{v}_{\omega,n}\,\boldsymbol{v}_{\omega,n}^{\top}\}$ is the autocorrelation matrix of $\boldsymbol{v}_{\omega,n}$ and $J_{\min} = E\{e_{\text{opt},\omega}^2\}$ the minimum MSE given by (4.33). The second term on the RHS of (4.52) is the excess MSE (EMSE), denoted by $J_{\text{EMSE}}(n)$. The study of the MSE behavior (4.52) requires a model for $\boldsymbol{C}_{v,\omega}(n)$, which highly depends on the transformation of the input signal $\boldsymbol{u}_n$ imposed by the multiple kernels. An analytical model for the behavior of $\boldsymbol{C}_{v,\omega}(n)$ was derived in [Parreira 2012]. It is assumed that the finite-order model provides a close enough approximation to the infinite-order model with minimum MSE, so that $E\{e_{\text{opt},\omega}^2\} \approx 0$. Post-multiplying (4.40) by its transpose, taking the expected value, and using the simplifying assumptions, we obtain the following recursion

$$\boldsymbol{C}_{v,\omega}(n+1) \approx \boldsymbol{C}_{v,\omega}(n) - \eta\left(\boldsymbol{R}_{\kappa\kappa}\,\boldsymbol{C}_{v,\omega}(n) + \boldsymbol{C}_{v,\omega}(n)\,\boldsymbol{R}_{\kappa\kappa}\right) + \eta^2\,\boldsymbol{T}_{\omega}(n) + \eta^2\,\boldsymbol{R}_{\kappa\kappa}\,J_{\min} \tag{4.53a}$$

with

$$\boldsymbol{T}_{\omega}(n) = E\left\{\boldsymbol{\kappa}_{\omega,n}\,\boldsymbol{\kappa}_{\omega,n}^{\top}\,\boldsymbol{v}_{\omega,n}\,\boldsymbol{v}_{\omega,n}^{\top}\,\boldsymbol{\kappa}_{\omega,n}\,\boldsymbol{\kappa}_{\omega,n}^{\top}\right\}. \tag{4.53b}$$

Evaluating (4.53b) is a challenging step in the analysis. As $\boldsymbol{\kappa}_{\omega,n}$ is a nonlinear transformation of a quadratic function of the Gaussian input vector $\boldsymbol{u}_n$, it is neither zero-mean nor Gaussian. In [Parreira 2012], for independent Gaussian-distributed dictionary elements, this leads to extensive calculations of up to eighth-order moments of $\boldsymbol{u}_n$. In Chapter 3, we have provided a greatly simplified alternative to this. However, both situations do not match the framework developed in this chapter, since the dictionary elements are now considered as preassigned a priori. The expectation of matrix $\boldsymbol{T}_{\omega}(n)$ for MI-MKLMS is derived as follows:

As the previous definitions in (4.42), we introduce the following notations

$$\begin{aligned} \hat{\ell} &= \bar{\ell} + (\ell - 1)M; \\ \hat{p} &= \bar{p} + (p - 1)M; \end{aligned} \tag{4.54}$$

where $1 \le \bar{\ell}, \bar{p} \le M$ and $1 \le \ell, p \le K$, following $1 \le \hat{\ell}, \hat{p} \le KM$. Using the CMIA to determine the $(\hat{i}, \hat{j})$-th element of $\boldsymbol{T}_{\omega}(n)$ in (4.53b) yields

$$[\boldsymbol{T}_{\omega}(n)]_{\hat{i},\hat{j}} \approx \sum_{\hat{\ell}=1}^{KM}\sum_{\hat{p}=1}^{KM} E_{\boldsymbol{u}}\left\{\kappa_{\omega,\hat{i}}(n)\,\kappa_{\omega,\hat{j}}(n)\,\kappa_{\omega,\hat{\ell}}(n)\,\kappa_{\omega,\hat{p}}(n)\right\} \cdot [\boldsymbol{C}_{v,\omega}(n)]_{\hat{\ell},\hat{p}} \tag{4.55}$$

where

$$\begin{aligned} \kappa_{\omega,\hat{q}}(n) &= \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_{\hat{q}}}) \\ &= \kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_{\bar{q}+(q-1)M}}). \end{aligned} \tag{4.56}$$

In matrix form, (4.55) can be rewritten as

$$[\boldsymbol{T}_{\omega}(n)]_{\hat{i},\hat{j}} \approx \operatorname{trace}\left\{\hat{\boldsymbol{K}}_{\omega}(\hat{i},\hat{j})\,\boldsymbol{C}_{v,\omega}(n)\right\}. \tag{4.57}$$

The $(\hat{\ell},\hat{p})$-th entry of $\hat{\boldsymbol{K}}_{\omega}(\hat{i},\hat{j})$ is given by

$$\begin{aligned}
[\hat{\boldsymbol{K}}_{\omega}(\hat{i},\hat{j})]_{\hat{\ell},\hat{p}} &= E_{\boldsymbol{u}}\left\{\kappa(\boldsymbol{u}_n,\boldsymbol{u}_{\omega_{\hat{i}}})\cdot\kappa(\boldsymbol{u}_n,\boldsymbol{u}_{\omega_{\hat{j}}})\cdot\kappa(\boldsymbol{u}_n,\boldsymbol{u}_{\omega_{\hat{\ell}}})\cdot\kappa(\boldsymbol{u}_n,\boldsymbol{u}_{\omega_{\hat{p}}})\right\}\\
&= E_{\boldsymbol{u}}\left\{\exp\left(-\frac{\varepsilon_1^2\|\boldsymbol{u}_n-\boldsymbol{u}_{\omega_{\hat{i}}}\|^2+\varepsilon_2^2\|\boldsymbol{u}_n-\boldsymbol{u}_{\omega_{\hat{j}}}\|^2+\varepsilon_3^2\|\boldsymbol{u}_n-\boldsymbol{u}_{\omega_{\hat{\ell}}}\|^2+\varepsilon_4^2\|\boldsymbol{u}_n-\boldsymbol{u}_{\omega_{\hat{p}}}\|^2}{2\epsilon_0^2}\right)\right\}\\
&= \exp\left(-\frac{1}{2\varepsilon_0^2}\|\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}\hat{\ell}\hat{p}}}\|^{(2)}\right)\cdot E_{\boldsymbol{u}}\left\{\exp\left(-\frac{1}{2\varepsilon_0^2}[\epsilon_0^2\|\boldsymbol{u}_n\|^2-2\,\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}\hat{\ell}\hat{p}}}^{\top}\boldsymbol{u}_n]\right)\right\}
\end{aligned} \tag{4.58}$$

where

$$\begin{aligned}
\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}\hat{\ell}\hat{p}}} &= \varepsilon_1^2\,\boldsymbol{u}_{\omega_{\hat{i}}}+\varepsilon_2^2\,\boldsymbol{u}_{\omega_{\hat{j}}}+\varepsilon_3^2\,\boldsymbol{u}_{\omega_{\hat{\ell}}}+\varepsilon_4^2\,\boldsymbol{u}_{\omega_{\hat{p}}}\\
\|\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}\hat{\ell}\hat{p}}}\|^{(2)} &= \varepsilon_1^2\,\|\boldsymbol{u}_{\omega_{\hat{i}}}\|^2+\varepsilon_2^2\,\|\boldsymbol{u}_{\omega_{\hat{j}}}\|^2+\varepsilon_3^2\,\|\boldsymbol{u}_{\omega_{\hat{\ell}}}\|^2+\varepsilon_4^2\,\|\boldsymbol{u}_{\omega_{\hat{p}}}\|^2
\end{aligned} \tag{4.59}$$

with the notations of ε_0, ε_1, ε_2, ε_3, ε_4 and ϵ_0 defined as

$$\begin{aligned}
\varepsilon_0 &= \xi_i\,\xi_j\,\xi_\ell\,\xi_p;\\
\varepsilon_1 &= \xi_j\,\xi_\ell\,\xi_p;\\
\varepsilon_2 &= \xi_i\,\xi_\ell\,\xi_p;\\
\varepsilon_3 &= \xi_i\,\xi_j\,\xi_p;\\
\varepsilon_4 &= \xi_i\,\xi_j\,\xi_\ell;\\
\epsilon_0 &= \left(\sum_{t=1}^{4}\varepsilon_t^2\right)^{\frac{1}{2}}.
\end{aligned} \tag{4.60}$$

Now setting $\boldsymbol{H}=\epsilon_0^2\boldsymbol{I}$, $\boldsymbol{b}=-2\,\bar{\boldsymbol{u}}_{\omega_{\bar{i}\bar{j}\bar{\ell}\bar{p}}}$ and $s=-1/2\varepsilon_0^2$ according to the moment generating function (4.36), (4.58) can be reformulated as

$$[\hat{\boldsymbol{K}}_{\omega}(\hat{i},\hat{j})]_{\hat{\ell},\hat{p}} = |\boldsymbol{I}+(\epsilon_0/\varepsilon_0)^2\boldsymbol{R}_{\boldsymbol{uu}}|^{-\frac{1}{2}}\cdot\exp\left(-\frac{1}{2\varepsilon_0^2\epsilon_0^2}\left[\epsilon_0^2\|\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}\hat{\ell}\hat{p}}}\|^{(2)}-\|\hat{\boldsymbol{u}}_{\omega_{\hat{i}\hat{j}\hat{\ell}\hat{p}}}\|^2_{(\boldsymbol{I}+(\varepsilon_0/\epsilon_0)^2\boldsymbol{R}_{\boldsymbol{uu}}^{-1})^{-1}}\right]\right). \tag{4.61}$$

4.3.4 Steady-state behavior

Once matrix $\hat{\boldsymbol{K}}_{\omega}$ is obtained from (4.61), $[\boldsymbol{T}_{\omega}(n)]_{i,j}$ can be computed with these known elements of $\hat{\boldsymbol{K}}_{\omega}$, and $\boldsymbol{C}_{v,\omega}(n)$. With these expressions, we complete the calculation of the second order model of the MI-MKLMS algorithm.

In order to facilitate the determination of the steady state of (4.53a), observing that it only involves linear operations on the elements of $\boldsymbol{C}_{v,\omega}(n)$, we now transform it into the form of product of matrix and vector as

$$\boldsymbol{c}_{v,\omega}(n+1) = \boldsymbol{G}_{\omega}\,\boldsymbol{c}_{v,\omega}(n)+\eta^2 J_{\min}\,\boldsymbol{r}_{\kappa,\omega} \tag{4.62}$$

with

$$\boldsymbol{G}_{\omega} = \boldsymbol{I} - \eta\,(\boldsymbol{G}_{\omega,1} + \boldsymbol{G}_{\omega,2}) + \eta^2\,\boldsymbol{G}_{\omega,3} \tag{4.63}$$

where vectors $\boldsymbol{c}_{v,\omega}(n)$ and $\boldsymbol{r}_{\kappa,\omega}$ are the lexicographic representations of matrix $\boldsymbol{C}_{v,\omega}(n)$ and $\boldsymbol{R}_{\kappa\kappa}$, respectively. The entries of $\boldsymbol{G}_{\omega}$ are defined as:

- $\boldsymbol{I}$ is the identity matrix with the dimension $(K^2M^2 \times K^2M^2)$.
- The matrix $\boldsymbol{G}_{\omega,1}$ corresponds to the product $\boldsymbol{C}_{v,\omega}(n)\,\boldsymbol{R}_{\kappa\kappa}$. It is a block matrix with $\boldsymbol{R}_{\kappa\kappa}$ along its diagonal, that is,

  $$\boldsymbol{G}_{\omega,1} = \boldsymbol{I}_{KM\times KM} \otimes \boldsymbol{R}_{\kappa\kappa} \tag{4.64}$$

  where operator $\otimes$ is Kronecker product.
- The matrix $\boldsymbol{G}_{\omega,2}$ corresponds to the product $\boldsymbol{R}_{\kappa\kappa}\,\boldsymbol{C}_{v,\omega}(n)$. It can be written as

  $$\boldsymbol{G}_{\omega,2} = \boldsymbol{R}_{\kappa\kappa} \otimes \boldsymbol{I}_{KM\times KM}. \tag{4.65}$$

- Any entry of $\boldsymbol{G}_{\omega,3}$ can be determined by

  $$[\boldsymbol{G}_{\omega,3}]_{\hat{i}+(\hat{j}-1)KM,\hat{\ell}+(\hat{p}-1)KM} = [\hat{\boldsymbol{K}}_{\omega}(\hat{i},\hat{j})]_{\hat{\ell},\hat{p}} \tag{4.66}$$

  with $1 \le \hat{i},\hat{j},\hat{\ell},\hat{p} \le KM$.

Note that $\boldsymbol{G}_{\omega,1}$ to $\boldsymbol{G}_{\omega,3}$ are all symmetric matrices, hence, $\boldsymbol{G}_{\omega}$ is also symmetric. Assuming convergence, the closed-form solution of (4.62) can be written as [Luenberger 1979]

$$\boldsymbol{c}_{v,\omega}(n) = \boldsymbol{G}_{\omega}^{n}\left[\boldsymbol{c}_{v,\omega}(0) - \boldsymbol{c}_{v,\omega}(\infty)\right] + \boldsymbol{c}_{v,\omega}(\infty) \tag{4.67}$$

where $\boldsymbol{c}_{v,\omega}(\infty)$ denotes the vector $\boldsymbol{c}_{v,\omega}(n)$ in steady-state by taking the limit $n \to \infty$, yielding

$$\begin{aligned}\boldsymbol{c}_{v,\omega}(\infty) &= \eta^2\,J_{\min}\,(\boldsymbol{I} - \boldsymbol{G}_{\omega})^{-1}\,\boldsymbol{r}_{\kappa,\omega} \\ &= \eta^2 J_{\min}\,[\eta(\boldsymbol{G}_{\omega,1} + \boldsymbol{G}_{\omega,2}) - \eta^2\boldsymbol{G}_{\omega,3}]^{-1}\,\boldsymbol{r}_{\kappa,\omega}.\end{aligned} \tag{4.68}$$

From (4.52), the steady-state MSE is given by

$$J_{\mathrm{MSE}}(\infty) = J_{\min} + \mathrm{trace}\,\{\boldsymbol{R}_{\kappa\kappa}\,\boldsymbol{C}_{v,\omega}(\infty)\} \tag{4.69}$$

where $J_{\mathrm{EMSE}}(\infty) = \mathrm{trace}\,\{\boldsymbol{R}_{\kappa\kappa}\,\boldsymbol{C}_{v,\omega}(\infty)\}$ is the steady-state EMSE of the MI-MKLMS algorithm.

Theorem 4.3.2 *(Mean-square stability) Assume CMIA holds. For any initial condition, given a dictionary $\boldsymbol{\omega}$, the second-order weight-error vector recursion (4.62) of the MI-MKLMS algorithm is mean-square stable if, and only if, the matrix $\boldsymbol{G}_{\omega}$ is stable. If the step size is sufficiently small so that the approximation:*

$$\boldsymbol{G}_{\omega} \approx \boldsymbol{R}_{\kappa\kappa}^{\top} \otimes \boldsymbol{R}_{\kappa\kappa}. \tag{4.70}$$

holds, the mean-square stability is ensured by the stability of $\boldsymbol{R}_{\kappa\kappa}$.

4.4 Simulation results and discussion

In this section, we consider two problems of nonlinear system identification already presented in Chapter 3 to validate the analytical models of the two so-called MKLMS algorithms. Without loss of generality, the total number of sub-kernels K was set to 2, and all the simulation results were averaged over 200 Monte Carlo runs. The nonnegative mixing parameters $[\beta_1, \beta_2]$ of the SI-MKLMS algorithm were set to [0.5, 0.5] in all cases.

4.4.1 Example 1

Consider the input sequence

$$u_n = \rho\, u_{n-1} + \sigma_u \sqrt{1-\rho^2}\, w_n \tag{4.71}$$

with w_n an i.i.d. noise signal driven by standard normal distribution. The input sequence u_n were zero-mean i.i.d. Gaussian noise with standard deviation $\sigma_u = 0.5$, and ρ was set to 0.5. The nonlinear system was defined as follows:

$$\begin{cases} y_n = 0.5\, u_n - 0.3\, u_{n-1} \\ d_n = y_n - 1.25\, y_n^2 + 0.25\, y_n^3 + z_n \end{cases} \tag{4.72}$$

where the output signal d_n was corrupted by a zero-mean white Gaussian noise z_n with standard deviation $\sigma_z = 0.1$. At each time instant n, the measured input data and reference signal were $\boldsymbol{u}_n = [u_n, u_{n-1}]^\top$ and d_n, respectively. The twenty-five dictionary elements were randomly pre-selected by the CS criterion with threshold $\delta_0 = 0.01$ and $\xi = 0.2$. The simulation conditions and the results of SI-MKLMS algorithm and MI-MKLMS algorithm are listed in Table 4.1 and Table 4.2. Figures 4.3 and 4.4 show that the model predictions of SI-MKLMS and MI-MKLMS consistently agree with Monte Carlo simulations in transient and steady-state stages.

Table 4.1: Summary of simulation results of the SI-MKLMS algorithm for Example 1.

Algorithm	M	η	ξ	$J_{\min}$ [dB]	$J_{\mathrm{ms}}(\infty)$ [dB]	$J_{\mathrm{EMSE}}(\infty)$ [dB]
KLMS	25	0.1	0.2	-15.46	-15.41	-34.92
			0.25	-17.52	-17.44	-35.03
SI-MKLMS			[0.2; 0.25]	**-16.69**	**-16.62**	-35.18

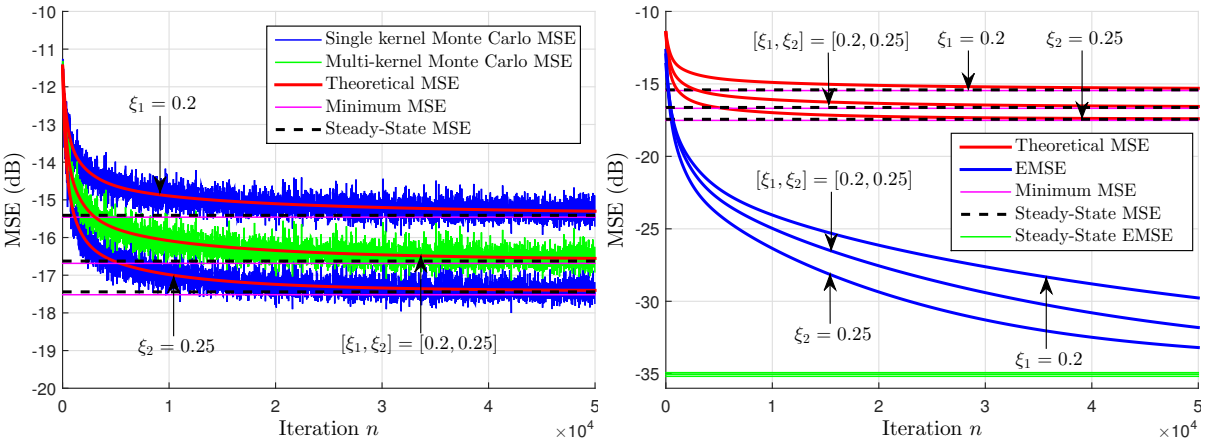
4.4.2 Example 2

As a second design example, we considered the nonlinear dynamic fund-flow control problem studied in [Vörös 2003] where the input signal was a sequence of statistically independent vectors generated by (4.71) with $\sigma_u = 0.25$ and $\rho = 0.5$. Consider a linear system with memory defined by

$$x_n = \boldsymbol{a}^\top \boldsymbol{u}_n - 0.2\, x_{n-1} + 0.35\, x_{n-2} \tag{4.73}$$

Table 4.2: Summary of simulation results of MI-MKLMS for Example 1.

Algorithm	M	η	ξ	$J_{\min}$ [dB]	$J_{\text{MSE}}(\infty)$ [dB]	$J_{\text{EMSE}}(\infty)$ [dB]
KLMS	25	0.1	0.2	-15.46	-15.41	-34.92
			0.25	-17.52	-17.44	-35.03
MI-MKLMS			[0.2; 0.25]	**-19.02**	**-18.90**	-34.28
KLMS	25	0.1	0.2	-15.48	-15.43	-34.94
			0.4	-19.76	-19.56	-33.07
MI-MKLMS			[0.2; 0.4]	-19.81	-19.56	-32.09



(a) Simulated and theoretical MSE curves.

(b) Theoretical MSE and EMSE curves.

Figure 4.3: Learning curves of SI-MKLMS algorithm for Example 1 ($\xi_1 = 0.2$, $\xi_2 = 0.25$).

where $\boldsymbol{a} = [1\ 0.5]^\top$ and a nonlinear Wiener function

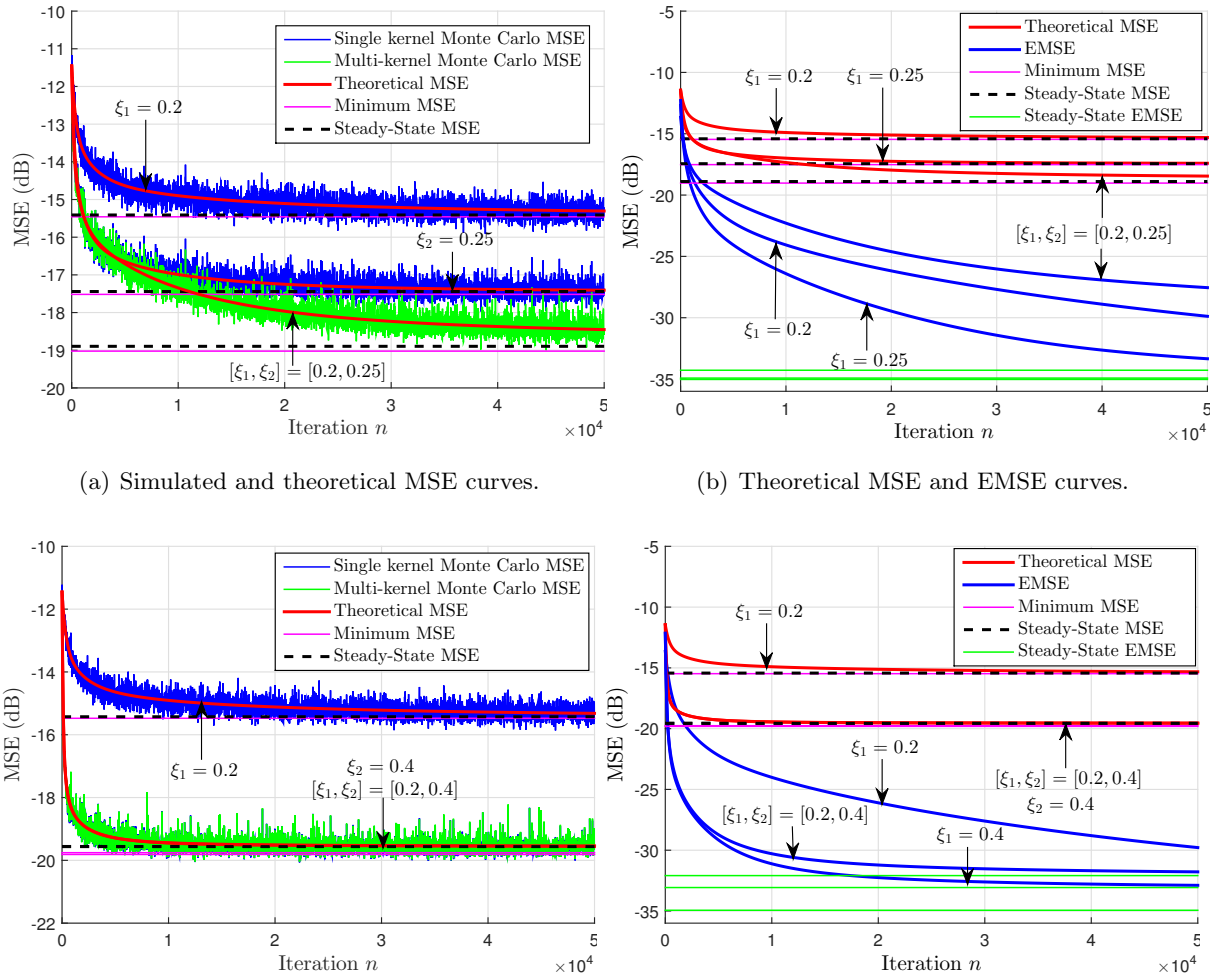
$$y(n) = \begin{cases} \dfrac{x_n}{3[0.1 + 0.9\,x_n^2]^{1/2}} & \text{for } x_n \geq 0 \\ \dfrac{-x_n^2[1 - \exp(0.7x_n)]}{3} & \text{for } x_n < 0, \end{cases} \tag{4.74}$$

$$d_n = y_n + z_n \tag{4.75}$$

where d_n is the output signal, corrupted by a zero-mean white Gaussian noise z_n with variance $\sigma_z = 0.01$. The initial condition $y_1 = 0$ was considered in this example. The forty-six dictionary elements were randomly generated by the CS criterion with threshold $\delta_0 = 1 \times 10^{-4}$ and $\xi = 0.05$. The simulation conditions and the results of SI-MKLMS algorithm and MI-MKLMS algorithm are listed in Table 4.3 and Table 4.4. The accuracy and effectiveness of theoretical models are illustrated in Figures 4.5 and 4.6.

4.4.3 Discussion

From the comparisons of simulation results, we can provide the following conclusions:



(a) Simulated and theoretical MSE curves.

(b) Theoretical MSE and EMSE curves.

(c) Simulated and theoretical MSE curves.

(d) Theoretical MSE and EMSE curves.

Figure 4.4: Learning curves of MI-MKLMS algorithm for Example 1 ($\xi_1 = 0.2$, $\xi_2 = 0.25$ for (a) and (b); $\xi_1 = 0.2$, $\xi_2 = 0.4$ for (c) and (d)).

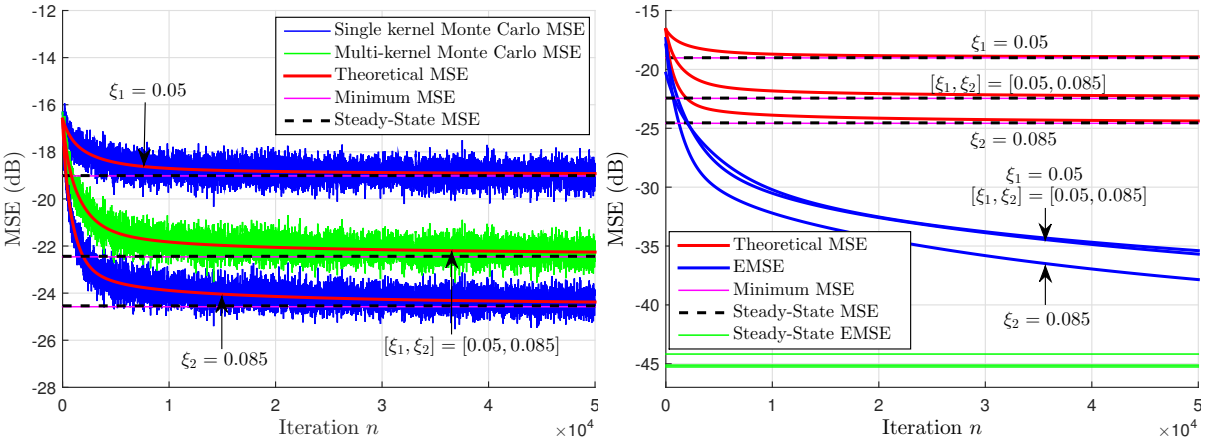
Table 4.3: Summary of simulation results of SI-MKLMS for Example 2.

Algorithm	M	η	ξ	$J_{\min}$ [dB]	$J_{\text{MSE}}(\infty)$ [dB]	$J_{\text{EMSE}}(\infty)$ [dB]
KLMS	46	0.05	0.05	-19.02	-19.01	-44.17
			0.085	-24.58	-24.54	-45.11
SI-MKLMS			[0.05; 0.085]	**-22.46**	**-22.44**	-45.27

1. The predictions of two multi-kernel analysis models perfectly match the Monte Carlo simulations as shown in Figures 4.3 to 4.6. When the total number of sub-kernels K is set 1, it is evident that both multi-kernel LMS algorithms reduce to the monokernel LMS algorithm. In other words, the monokernel LMS algorithm is a special case of

Table 4.4: Summary of simulation results of MI-MKLMS for Example 2.

Algorithm	M	η	ξ	$J_{\min}$ [dB]	$J_{\text{MSE}}(\infty)$ [dB]	$J_{\text{EMSE}}(\infty)$ [dB]
KLMS	46	0.05	0.05	-19.03	-19.01	-44.18
			0.085	-24.53	-24.49	-45.06
MI-MKLMS			[0.05; 0.085]	**-26.39**	**-26.33**	-45.59
KLMS	46	0.05	0.05	-19.02	-19.00	-44.17
			0.15	-28.65	-28.53	-44.19
MI-MKLMS			[0.05; 0.15]	-28.71	-28.58	-43.77



(a) Simulated and theoretical MSE curves.

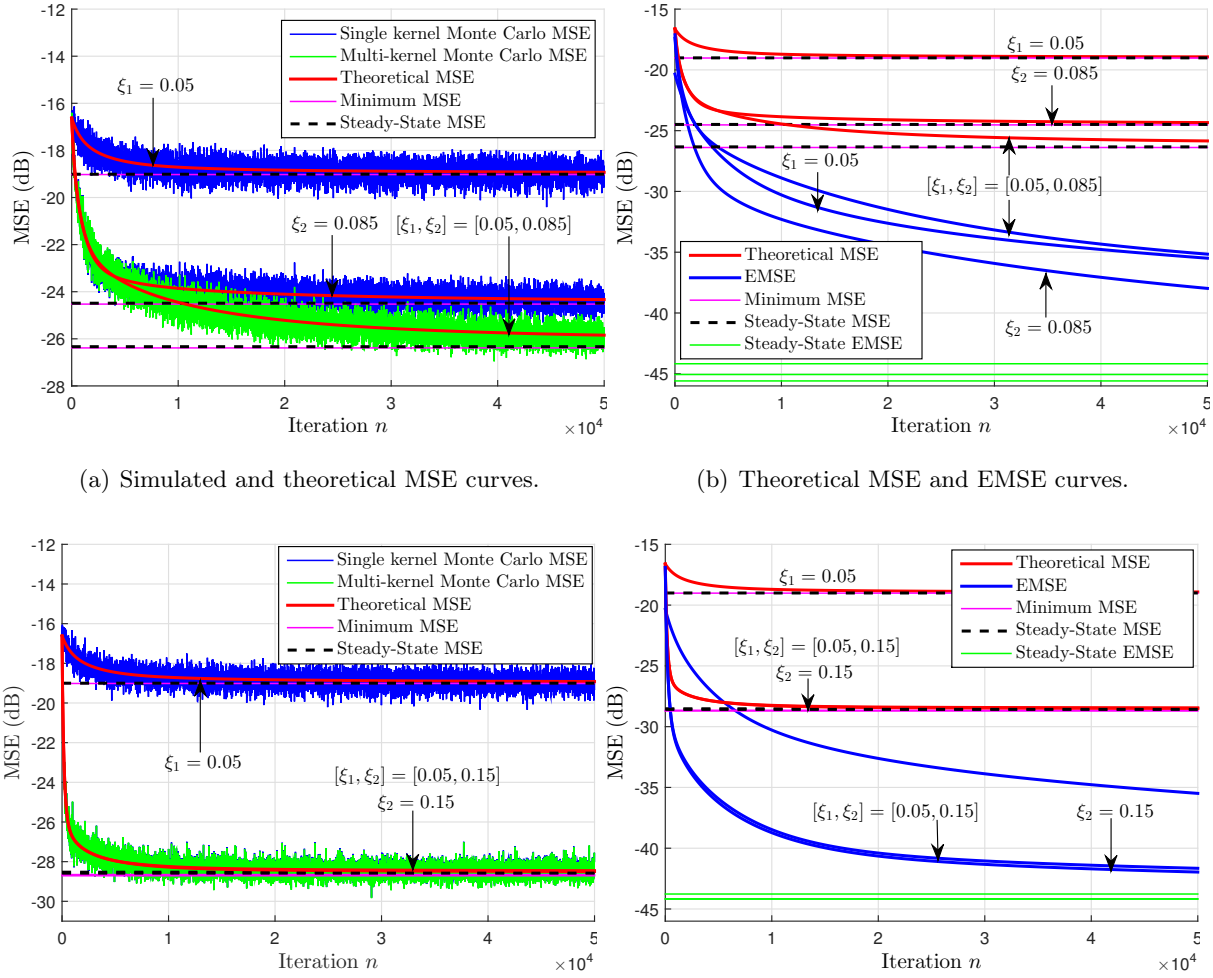
(b) Theoretical MSE and EMSE curves.

Figure 4.5: Learning curves of SI-MKLMS algorithm for Example 2 ($\xi_1 = 0.05$, $\xi_2 = 0.085$).

the MKLMS algorithms.

2. SI-MKLMS algorithm allows to combine the capability of sub-kernels into an integrated single kernel. Hence, it is possible to trade off the convergence speed and MSE of monokernel LMS algorithm by tuning the weight coefficients β_k, which can be optimized by gradient descent method at each iteration [Rakotomamonjy 2008, Chen 2013c]. Therefore, the SI-MKLMS is for instance appropriate in case of lack of prior statistical information on input data.

3. When the bandwidths of sub-kernels uniformly cover the interval between minimum and maximum bandwidth of single Gaussian kernel [$\xi_{\min}$ $\xi_{\max}$], the performance of MI-MKLMS algorithm are superior to SI-MKLMS and corresponding monokernel LMS, because of its faster convergence speed and lower MSE as shown in Figures 4.4 and 4.6. It is an effective and feasible method that cascading more sub-kernels, i.e., increasing the number K to cover the larger range of required bandwidth. Nevertheless, this increases the computational cost.



(a) Simulated and theoretical MSE curves.

(b) Theoretical MSE and EMSE curves.

(c) Simulated and theoretical MSE curves.

(d) Theoretical MSE and EMSE curves.

Figure 4.6: Learning curves of MI-MKLMS algorithm for Example 2 ($\xi_1 = 0.05$, $\xi_2 = 0.085$ for (a) and (b); $\xi_1 = 0.05$, $\xi_2 = 0.15$ for (c) and (d)).

4.5 Conclusion

In this chapter we presented the two of multi-kernel LMS schemes and discussed their characteristics. Then we analyzed the convergence behavior of the MI-MKLMS algorithm with pre-tuned dictionary. Finally, numerical simulations validated the accuracy of the theoretical models of the MI-MKLMS. We further compared the performance of SI-MKLMS and MI-MKLMS algorithms.

Chapter 5

Complex kernel adaptive filtering algorithm

Contents

5.1 Introduction

Kernel-based adaptive filtering algorithms for complex data have recently attracted attention since they ensure phase processing. This is of importance for applications in communication, radar, sonar, etc. The concept and properties of Wirtinger's calculus was formally introduced to calculate the gradients of the non-holomorphic cost function defined in general complex RKHS [Bouboulis 2010]. Afterwards, two schemes of complexified real kernel LMS and pure complex kernel LMS were proposed in [Bouboulis 2011]. A direct extension of the derivations in [Parreira 2012] was proposed in [Thomas 2013] to analyze the convergence behavior of complex KLMS algorithm (CKLMS). The augmented complex LMS (ACLMS) algorithm was presented in [Mandic 2009, Kung 2009], and its nonlinear counterpart augmented normalized complex KLMS (ANCKLMS) approach was described in [Bouboulis 2012] and [Tobar 2012a]. These works show that the augmented complex-valued algorithms provide significantly improved performance compared with the usual complex-valued algorithms.

The aim of this chapter is to study the convergence behavior of the augmented complex Gaussian KLMS algorithm. First, we present two complex monokernel adaptive filtering algorithms: complexified real kernel LMS and pure complex kernel LMS. Then, some definitions and a general framework are introduced for pure complex multikernel adaptive filtering algorithms. This framework relies on multikernel adaptive filters that has previously been derived for use with real-valued data in [Yukawa 2012, Tobar 2014a, Gao 2014a, Pokharel 2013]. Next, we derive models for the convergence behavior in the mean and mean-square sense of the ACKLMS algorithm with complex Gaussian kernels. Finally, the accuracy of these models is checked with simulation results.

5.2 Complex monokernel adaptive filtering algorithms

In this section, we shall briefly introduce two existing complex monokernel adaptive algorithms.

5.2.1 Complexified kernel LMS algorithm

Consider the complex input/output sequence $\{(\boldsymbol{u}(n), d(n))\}_{n=1}^{N}$ with $\boldsymbol{u}(n) \in \mathbb{U}$ and $d(n) \in \mathbb{C}$, where $\mathbb{U}$ is a compact of $\mathbb{C}^L$. The complex input vector can be expressed in the form

$$\begin{aligned} \boldsymbol{u}(n) &= \sqrt{1-\rho^2}\,\boldsymbol{u}_{\text{re}}(n) + \mathrm{i}\rho\,\boldsymbol{u}_{\text{im}}(n) \\ &= \boldsymbol{u}_I(n) + \mathrm{i}\,\boldsymbol{u}_Q(n) \end{aligned} \tag{5.1}$$

where the subscripts I and Q denote "in-phase" and "quadrature" components, and $\mathrm{i} = \sqrt{-1}$. The sequence $\boldsymbol{u}_{\text{re}}(n)$ (resp., $\boldsymbol{u}_{\text{im}}(n)$) is supposed to be zero-mean, independent, and identically distributed according to a real-valued Gaussian distribution. The entries of each input vector $\boldsymbol{u}_{\text{re}}(n)$ (resp., $\boldsymbol{u}_{\text{im}}(n)$) can, however, be correlated. In addition, the sequences $\boldsymbol{u}_{\text{re}}(n)$ and $\boldsymbol{u}_{\text{im}}(n)$ are assumed to be independent. This implies that $E\{\boldsymbol{u}(n-i)\boldsymbol{u}^H(n-j)\} = \boldsymbol{0}$ for $i \neq j$, where the operator $(\cdot)^H$ denotes Hermitian transpose. The circularity of input data is controlled by parameter ρ. Setting the parameter $\rho = \sqrt{2}/2$ results in a circular input, while ρ approaching to 0 or 1 leads to a highly non-circular input.

The essence of the so-called complexified kernel LMS is actually the generalized real mono-kernel LMS approach applied stacking the real-valued input vectors $\boldsymbol{u}_I(n)$ and $\boldsymbol{u}_Q(n)$ as

$$\tilde{\boldsymbol{u}}(n) = \left[\boldsymbol{u}_I^{\top}(n)\ \ \boldsymbol{u}_Q^{\top}(n)\right]^{\top} \in \mathbb{R}^{2L\times 1}. \tag{5.2}$$

The block diagram of the complexified kernel LMS algorithm is depicted in Figure 5.1. The additive noise $z(n)$ is assumed to be white and zero-mean, with variance σ_z^2. Let the input space $\mathcal{U}^2$ be a compact of $\mathbb{R}^{2L}$, $\kappa_{\mathbb{R}} : \mathcal{U}^2 \times \mathcal{U}^2 \to \mathcal{H}^2$ be a real-valued reproducing kernel, and $\left(\mathcal{H}^2, \langle\cdot,\cdot\rangle\right)_{\mathcal{H}^2}$ be the induced real RKHS with its inner product. The real valued kernel function can be chosen, e.g., as the Gaussian kernel defined in Chapter 2

$$\kappa_{\mathbb{R}}(\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}}) = \exp\left(-\frac{\|\tilde{\boldsymbol{u}} - \tilde{\boldsymbol{v}}\|_2^2}{2\xi^2}\right) \tag{5.3}$$

with $\tilde{\boldsymbol{u}}, \tilde{\boldsymbol{v}} \in \mathbb{R}^{2L}$, and ξ the kernel bandwidth.

At iteration n, upon the arrival of new complex data pair $\{\boldsymbol{u}(n), d(n)\}$, one of the following alternatives holds based on the CS criterion. If $\kappa(\cdot, \boldsymbol{u}(n))$ does not satisfy the CS rule, the dictionary remains unchanged. On the other hand, if the condition (3.1) is met, the input $\boldsymbol{u}(n)$ is added to the dictionary where it is then denoted by $\kappa(\cdot, \boldsymbol{u}(\omega_{M+1}))$. Transforming the complex data into real-valued data and applying monokernel LMS, we obtain the complexified kernel LMS algorithm described as follows [Bouboulis 2011]

- **Rejection case:** $\max_{m=1,\cdots,M} \left|\kappa_{\mathbb{R}}(\tilde{\boldsymbol{u}}(n), \tilde{\boldsymbol{u}}(\omega_m))\right| > \delta_0$

$$\boldsymbol{\alpha}(n+1) = \boldsymbol{\alpha}(n) + \eta\, e(n)\, \boldsymbol{\kappa}_{\mathbb{R}}(\boldsymbol{\omega}_n) \tag{5.4}$$

- **Acceptance case:** $\max_{m=1,\cdots,M} \left|\kappa_{\mathbb{R}}(\tilde{\boldsymbol{u}}(n), \tilde{\boldsymbol{u}}(\omega_m))\right| \leq \delta_0$

$$\boldsymbol{\alpha}(n+1) = \begin{bmatrix} \boldsymbol{\alpha}(n) \\ 0 \end{bmatrix} + \eta\, e(n)\, \boldsymbol{\kappa}_{\mathbb{R}}(\boldsymbol{\omega}_n) \tag{5.5}$$

where η is a positive step-size, and $e(n) = \tilde{d}(n) - \boldsymbol{\alpha}^{\top}(n)\boldsymbol{\kappa}_{\mathbb{R}}(\boldsymbol{\omega}_n)$ is the estimation error with

$$\boldsymbol{\kappa}_{\mathbb{R}}(\boldsymbol{\omega}_n) = \left[\kappa_{\mathbb{R}}(\tilde{\boldsymbol{u}}(n), \tilde{\boldsymbol{u}}(\omega_1)), \cdots, \kappa_{\mathbb{R}}(\tilde{\boldsymbol{u}}(n), \tilde{\boldsymbol{u}}(\omega_M))\right]^{\top}. \tag{5.6}$$

The estimated output at iteration n is given by

$$\hat{d}(n) = \sum_{m=1}^{M} \alpha_m(n)\ \kappa_{\mathbb{R}}(\tilde{\boldsymbol{u}}(n), \tilde{\boldsymbol{u}}(\omega_m)). \tag{5.7}$$

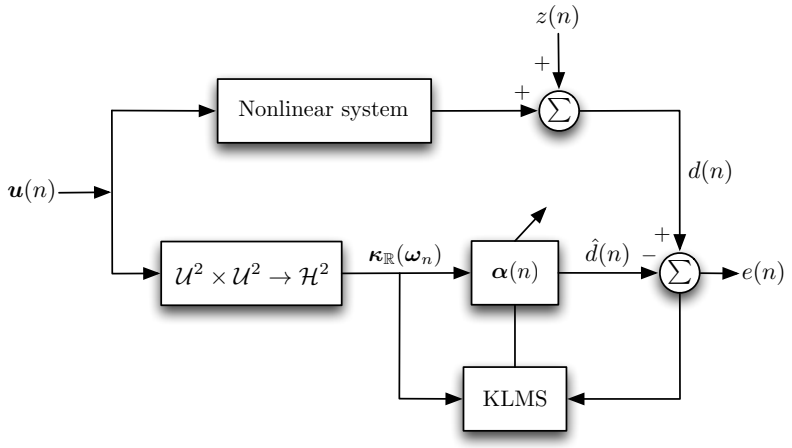


Figure 5.1: Block diagram of complexified kernel LMS algorithm.

5.2.2 Pure complex kernel LMS algorithm

Let $\kappa_{\mathbb{C}} : \mathbb{U} \times \mathbb{U} \to \mathbb{C}$ be a complex reproducing kernel. We denote by $(\mathbb{H}, \langle\cdot,\cdot\rangle_{\mathbb{H}})$ the induced complex RKHS. Complex reproducing kernels include in particular the Szego kernel, the

Bergman kernel, and the pure complex Gaussian kernel. The latter is the extension of the Gaussian kernel for complex arguments. The pure complex Gaussian kernel is defined as [Steinwart 2006]

$$\kappa_{\mathbb{C}}(\boldsymbol{u},\boldsymbol{v}) = \exp\left(-\frac{\sum_{\ell=1}^{L}(u_\ell - v_\ell^*)^2}{2\xi^2}\right) \tag{5.8}$$

with u_ℓ and v_ℓ the ℓ-th entries of $\boldsymbol{u},\boldsymbol{v}\in\mathbb{C}^L$. The parameter $\xi>0$ is the kernel bandwidth and $(\cdot)^*$ denotes the conjugate operator. The conjugate of kernel $\kappa_{\mathbb{C}}(\boldsymbol{u},\boldsymbol{v})$ is defined by

$$\kappa_{\mathbb{C}}^{\star}(\boldsymbol{u},\boldsymbol{v}) = \exp\left(-\frac{\sum_{\ell=1}^{L}(v_\ell - u_\ell^*)^2}{2\xi^2}\right). \tag{5.9}$$

Note that $(\cdot)^\star$ is defined on kernels and should not be confounded with the complex conjugate $(\cdot)^*$. We shall focus on the above complex Gaussian kernel in the sequel.

Calculating the gradient $\nabla_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha})$ enables to formulate steep-descent algorithms. For detailed procedures and formulations, please refer to [Bouboulis 2011]. According to the basic scheme of conventional complex LMS we can obtain the nonlinear pure complex kernel LMS algorithm:

- **Rejection case:** $\max_{m=1,\cdots,M}\left|\kappa_{\mathbb{C}}(\boldsymbol{u}(n),\boldsymbol{u}(\omega_m))\right| > \delta_0$

$$\boldsymbol{\alpha}(n+1) = \boldsymbol{\alpha}(n) + \eta\, e^*(n)\,\boldsymbol{\kappa}_{\mathbb{C}}(\boldsymbol{\omega}_n) \tag{5.10}$$

- **Acceptance case:** $\max_{m=1,\cdots,M}\left|\kappa_{\mathbb{C}}(\boldsymbol{u}(n),\boldsymbol{u}(\omega_m))\right| \le \delta_0$

$$\boldsymbol{\alpha}(n+1) = \begin{bmatrix}\boldsymbol{\alpha}(n)\\ 0\end{bmatrix} + \eta\, e^*(n)\,\boldsymbol{\kappa}_{\mathbb{C}}(\boldsymbol{\omega}_n) \tag{5.11}$$

where $e(n) = d(n) - \boldsymbol{\alpha}^H(n)\boldsymbol{\kappa}_{\mathbb{C}}(\boldsymbol{\omega}_n)$ is the estimation error and

$$\boldsymbol{\kappa}_{\mathbb{C}}(\boldsymbol{\omega}_n) = \left[\kappa_{\mathbb{C}}(\boldsymbol{u}(n),\boldsymbol{u}(\omega_1)),\cdots,\kappa_{\mathbb{C}}(\boldsymbol{u}(n),\boldsymbol{u}(\omega_M))\right]^\top. \tag{5.12}$$

The estimated output at iteration n is given by

$$\hat{d}(n) = \sum_{m=1}^{M}\alpha_m^*(n)\;\kappa_{\mathbb{C}}(\boldsymbol{u}(n),\boldsymbol{u}(\omega_m)). \tag{5.13}$$

The basic principles of pure complex kernel LMS algorithm were illustrated in Figure 5.2. We shall show in the next section that pure complex kernel LMS algorithm is a particular case of the augmented complex kernel LMS algorithm.

5.3 Complex multikernel adaptive filtering

5.3.1 The framework

Let $\{\kappa_{\mathbb{C},k}\}_{k=1}^K$ be the family of candidate complex kernels, and $\mathbb{H}_k$ the RKHS defined by each $\kappa_{\mathbb{C},k}$. Consider the space $\mathbb{H}$ of multidimensional mappings

$$\begin{aligned}\Phi:\quad &\mathbb{C}\longrightarrow\mathbb{C}^K\\ &\boldsymbol{u}\longmapsto\Phi(\boldsymbol{u}) = \mathrm{col}\big\{\varphi_1(\boldsymbol{u}),\ldots,\varphi_K(\boldsymbol{u})\big\}\end{aligned} \tag{5.14}$$

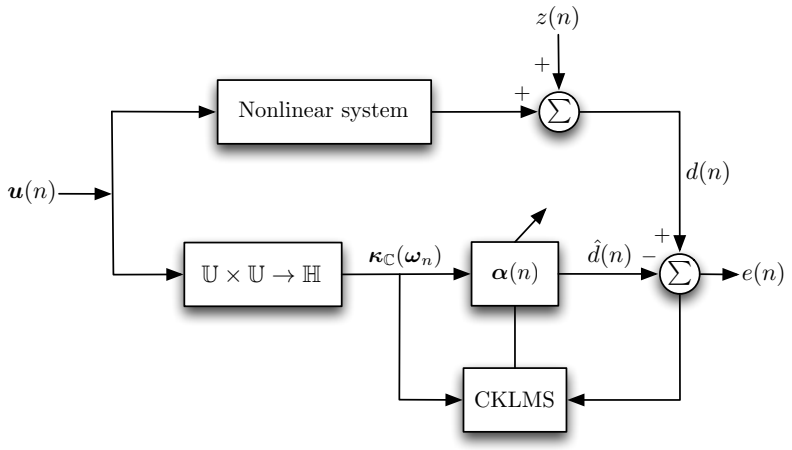


Figure 5.2: Block diagram of pure complex kernel LMS algorithm.

with $\varphi_k \in \mathbb{H}_k$ and $\mathrm{col}\{\cdot\}$ the operator that stacks its arguments on top of each other. Let $\langle \cdot, \cdot \rangle_{\mathbb{H}}$ be the inner product in $\mathbb{H}$ defined as

$$\langle \Phi, \Phi' \rangle_{\mathbb{H}} = \textstyle\sum_{k=1}^{K} \langle \varphi_k, \varphi'_k \rangle_{\mathbb{H}_k}. \tag{5.15}$$

The space $\mathbb{H}$ equipped with the inner product $\langle \cdot, \cdot \rangle_{\mathbb{H}}$ is a Hilbert space as $(\mathbb{H}_k, \langle \cdot, \cdot \rangle_{\mathbb{H}_k})$ is a complex Hilbert space for all k. We can then define the vector-valued representer of evaluation $\boldsymbol{\kappa}_{\mathbb{H}}(\cdot, \boldsymbol{u})$ such that

$$\Phi(\boldsymbol{u}) = [\Phi, \boldsymbol{\kappa}_{\mathbb{H}}(\cdot, \boldsymbol{u})] \tag{5.16}$$

with $\boldsymbol{\kappa}_{\mathbb{H}}(\cdot, \boldsymbol{u}) = \mathrm{col}\{\kappa_{\mathbb{C},1}(\cdot, \boldsymbol{u}), \ldots, \kappa_{\mathbb{C},K}(\cdot, \boldsymbol{u})\}$ and $[\cdot, \cdot]$ the entrywise inner product. This yields the following reproducing property

$$\boldsymbol{\kappa}_{\mathbb{H}}(\boldsymbol{u}, \boldsymbol{v}) = [\boldsymbol{\kappa}_{\mathbb{H}}(\cdot, \boldsymbol{u}), \boldsymbol{\kappa}_{\mathbb{H}}(\cdot, \boldsymbol{v})]. \tag{5.17}$$

Let $\Psi = \mathrm{col}\{\psi_1, \ldots, \psi_K\}$ be a vector-valued function in space $\mathbb{H}$, and let $\psi = \sum_{k=1}^{K} \psi_k$ with $\psi_k \in \mathbb{H}_k$ be the scalar-valued function that sums the entries of Ψ, namely, $\psi = \mathbf{1}_K^\top \Psi$ with $\mathbf{1}_K$ the all-one column vector of length K.

Given an input-output sequence $\{(d(n), \boldsymbol{u}(n))\}_{n=1}^{N}$, we aim at estimating a multidimensional function Ψ in $\mathbb{H}$ that minimizes the regularized least-square error

$$\min_{\Psi \in \mathbb{H}} J(\Psi) = \sum_{n=1}^{N} \left| d(n) - \mathbf{1}_K^\top \Psi(\boldsymbol{u}(n)) \right|^2 + \lambda \|\mathbf{1}_K^\top \Psi\|_{\mathbb{H}}^2 \tag{5.18}$$

with $\lambda \geq 0$ a regularization constant. By virtue of the generalized multidimensional representer theorem, the optimum function Ψ can be written as

$$\Psi(\cdot) = \mathrm{col}\Big\{ \sum_{n=1}^{N} \alpha_{n,k}^{*} \, \kappa_{\mathbb{C},k}(\cdot, \boldsymbol{u}(n)) \Big\}_{k=1}^{K}. \tag{5.19}$$

For simplicity, without loss of generality, we shall omit the regularization term in problem (5.18), which can be reformulated as

$$\min_{\boldsymbol{\alpha}} J(\boldsymbol{\alpha}) = \sum_{n=1}^{N} \left| d(n) - \sum_{k=1}^{K} \boldsymbol{\alpha}_k^H \boldsymbol{\kappa}_{\mathbb{C},k}(n) \right|^2 \tag{5.20}$$

where $\boldsymbol{\alpha} = \text{col}\{\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_K\}$ with $\boldsymbol{\alpha}_k = (\alpha_{1,k}, \ldots, \alpha_{N,k})^\top$ is the unknown weight vector, and $\boldsymbol{\kappa}_{\mathbb{C},k}(n)$ is the $N \times 1$ kernelized input vector with j-th entry $\kappa_{\mathbb{C},k}(\boldsymbol{u}(j), \boldsymbol{u}(n))$. Calculating the directional derivative of $J(\boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$ by Wirtinger's calculus yields

$$\partial_{\boldsymbol{\alpha}_k} J(\boldsymbol{\alpha}) = -2 \sum_{n=1}^{N} e^*(n)\, \boldsymbol{\kappa}_{\mathbb{C},k}(\cdot, \boldsymbol{u}(n)). \tag{5.21}$$

where $e(n) = d(n) - \sum_{k=1}^{K} \boldsymbol{\alpha}_k^H \boldsymbol{\kappa}_{\mathbb{C},k}(n)$. Approximating (5.21) by its instantaneous estimate $\partial_{\boldsymbol{\alpha}_k} J(\boldsymbol{\alpha}) \approx -2\, e^*(n)\, \kappa_{\mathbb{C},k}(\cdot, \boldsymbol{u}(n))$, we obtain the stochastic gradient descent algorithm:

$$\boldsymbol{\alpha}(n+1) = \boldsymbol{\alpha}(n) + \eta\, e^*(n)\, \boldsymbol{\kappa}_{\mathbb{H}}(n) = \sum_{i=1}^{n} \eta\, e^*(i)\, \boldsymbol{\kappa}_{\mathbb{H}}(i) \tag{5.22}$$

with $\boldsymbol{\kappa}_{\mathbb{H}}(n) = \text{col}\{\boldsymbol{\kappa}_{\mathbb{C},k}(n)\}_{k=1}^{K}$ the complex kernelized input vector, and $e(n) = d(n) - \boldsymbol{\alpha}^H(n)\, \boldsymbol{\kappa}_{\mathbb{H}}(n)$ the estimation error. Finally, the optimal function is of the form

$$\psi(\cdot) = \sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{n,k}^*\, \kappa_{\mathbb{C},k}(\cdot, \boldsymbol{u}(n)). \tag{5.23}$$

5.3.2 Augmented complex kernel least-mean-squared algorithm

In order to overcome the problem of the increasing amount n of observations in an online context, a fixed-size model is usually adopted:

$$\psi(\cdot) = \sum_{m=1}^{M} \sum_{k=1}^{K} \alpha_{m,k}^*\, \kappa_{\mathbb{C},k}(\cdot, \boldsymbol{u}(\omega_m)) \tag{5.24}$$

where $\boldsymbol{\omega} \triangleq \{\kappa_{\mathbb{H}}(\cdot, \boldsymbol{u}(\omega_m))\}_{m=1}^{M}$ is the so-called dictionary, and M its length. Limiting the number of monokernel filters to $K = 2$, and setting the two kernels to (5.8)– (5.9), the augmented complex kernel least-mean-squared (ACKLMS) algorithm based on model (5.24) is given by (See [Bouboulis 2012] for an introduction to ACKLMS):

$$\begin{aligned} \hat{d}(n) &= \sum_{m=1}^{M} \left[\alpha_{1,m}^*(n)\, \kappa_{\mathbb{C}}\big(\boldsymbol{u}(n), \boldsymbol{u}(\omega_m)\big) + \alpha_{2,m}^*(n)\, \kappa_{\mathbb{C}}^{\star}\big(\boldsymbol{u}(n), \boldsymbol{u}(\omega_m)\big) \right] \\ &= \boldsymbol{\alpha}^H(n)\, \boldsymbol{\kappa}_{\mathbb{H},\omega}(n). \end{aligned} \tag{5.25}$$

The block diagram of ACKLMS algorithm is depicted in Figure 5.3. The ACKLMS algorithm can be viewed as a complex Gaussian bi-kernel case of the complex multikernel algorithm [Bouboulis 2012, Tobar 2012a]. It can be expected that the ACKLMS algorithm outperforms the existing CKLMS algorithms due to the flexibility of complex multi-kernels.

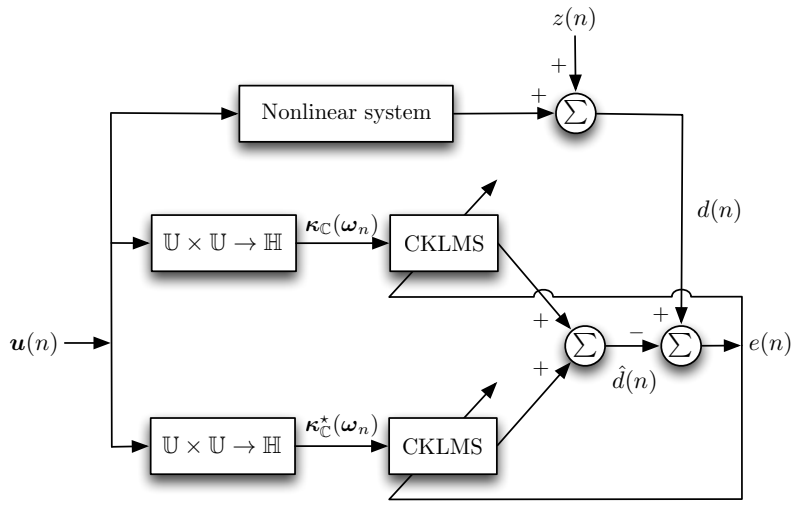


Figure 5.3: Block diagram of ACKLMS algorithm.

5.4 Stochastic behavior analysis of ACKLMS algorithm

We shall now study the transient and steady-state of the mean-square error conditionally to dictionary $\boldsymbol{\omega}$ of the complex Gaussian bi-kernel LMS algorithm, that is,

$$E\left\{|e(n)|^2 \,|\, \boldsymbol{\omega}\right\} = \int_{\mathbb{U}\times\mathbb{C}} |e(n)|^2 \, d\rho(\boldsymbol{u}(n), d(n) \,|\, \boldsymbol{\omega}) \tag{5.26}$$

with $e(n) = d(n) - \hat{d}(n)$ and ρ a Borel probability measure. We shall use the subscript ω for quantities conditioned on dictionary $\boldsymbol{\omega}$. Given $\boldsymbol{\omega}$, the estimation error at time instant n is given by

$$e_{\omega}(n) = d(n) - \hat{d}_{\boldsymbol{\omega}}(n) \tag{5.27}$$

with $\hat{d}_{\boldsymbol{\omega}}(n) = \hat{d}(n)|\boldsymbol{\omega}$. Multiplying $e_{\omega}(n)$ by its conjugate and taking the expected value yields the MSE

$$J_{\text{ms}} = E\{|d(n)|^2\} - 2\,\text{Re}\big(\boldsymbol{p}^H_{\kappa d,\omega}\boldsymbol{\alpha}_{\omega}(n)\big) + \boldsymbol{\alpha}^H_{\omega}(n)\boldsymbol{R}_{\kappa\kappa}\boldsymbol{\alpha}_{\omega}(n) \tag{5.28}$$

with $\boldsymbol{R}_{\kappa\kappa} = E\big\{\boldsymbol{\kappa}_{\mathbb{H},\omega}(n)\boldsymbol{\kappa}^H_{\mathbb{H},\omega}(n)|\boldsymbol{\omega}\big\}$ the correlation matrix of input data, and $\boldsymbol{p}_{\kappa d,\omega} = E\left\{\boldsymbol{\kappa}_{\mathbb{H},\omega}(n)d^*(n)|\boldsymbol{\omega}\right\}$ the cross-correlation vector between $\boldsymbol{\kappa}_{\mathbb{H},\omega}(n)$ and $d(n)$. As $\boldsymbol{R}_{\kappa\kappa}$ is positive definite, the optimum weight vector is given by

$$\boldsymbol{\alpha}_{\text{opt},\omega} = \arg\min_{\boldsymbol{\alpha}_{\omega}} J_{\text{ms}}(\boldsymbol{\alpha}_{\omega}) = \boldsymbol{R}^{-1}_{\kappa\kappa}\,\boldsymbol{p}_{\kappa d,\omega} \tag{5.29}$$

and the minimum MSE is

$$J_{\text{min}} = E\left\{|d(n)|^2\right\} - \boldsymbol{p}^H_{\kappa d,\omega}\,\boldsymbol{R}^{-1}_{\kappa\kappa}\,\boldsymbol{p}_{\kappa d,\omega}. \tag{5.30}$$

5.4.1 Mean weight error analysis

The weight update of the ACKLMS algorithm is given by

$$\boldsymbol{\alpha}_{\omega}(n+1) = \boldsymbol{\alpha}_{\omega}(n) + \eta\, e^{*}_{\omega}(n)\, \boldsymbol{\kappa}_{\mathbb{H},\omega}(n). \tag{5.31}$$

Let $\boldsymbol{v}_{\omega}(n)$ be the weight error vector defined as

$$\boldsymbol{v}_{\omega}(n) = \boldsymbol{\alpha}_{\omega}(n) - \boldsymbol{\alpha}_{\text{opt},\omega}. \tag{5.32}$$

The weight error vector update equation is then given by

$$\boldsymbol{v}_{\omega}(n+1) = \boldsymbol{v}_{\omega}(n) + \eta\, e^{*}_{\omega}(n)\, \boldsymbol{\kappa}_{\mathbb{H},\omega}(n). \tag{5.33}$$

The error (5.27) is consequently rewritten as

$$e_{\omega}(n) = d(n) - \boldsymbol{\kappa}^{H}_{\mathbb{H},\omega}(n)\boldsymbol{v}_{\omega}(n) - \boldsymbol{\kappa}^{H}_{\mathbb{H},\omega}(n)\boldsymbol{\alpha}_{\text{opt},\omega}. \tag{5.34}$$

Substituting (5.34) into (5.33) yields

$$\boldsymbol{v}_{\omega}(n+1) = \boldsymbol{v}_{\omega}(n) + \eta\big[d^{*}(n)\boldsymbol{\kappa}_{\mathbb{H},\omega}(n) - \boldsymbol{\kappa}^{H}_{\mathbb{H},\omega}(n)\boldsymbol{v}_{\omega}(n)\boldsymbol{\kappa}_{\mathbb{H},\omega}(n) - \boldsymbol{\kappa}^{H}_{\mathbb{H},\omega}(n)\boldsymbol{\alpha}_{\text{opt},\omega}\boldsymbol{\kappa}_{\mathbb{H},\omega}(n)\big]. \tag{5.35}$$

Taking the expected value of (5.35), using the CMIA hypothesis introduced in the previous chapter, and (5.29), we get the mean weight error model:

$$E\left\{\boldsymbol{v}_{\omega}(n+1)\right\} = (\boldsymbol{I} - \eta\, \boldsymbol{R}_{\kappa\kappa})E\left\{\boldsymbol{v}_{\omega}(n)\right\}. \tag{5.36}$$

The (i,j)-th entry of matrix $\boldsymbol{R}_{\kappa\kappa}$ is given by

$$[\boldsymbol{R}_{\kappa\kappa}]_{i,j} = E\big\{\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n), \boldsymbol{u}(\omega_i)\big)\left[\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n), \boldsymbol{u}(\omega_j)\big)\right]^{*}\big\} \tag{5.37}$$

with the complex Gaussian bi-kernel $\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n), \boldsymbol{u}(\omega_m)\big)$ given by

$$\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n), \boldsymbol{u}(\omega_m)\big) = \begin{cases} \kappa_{\mathbb{C}}\big(\boldsymbol{u}(n), \boldsymbol{u}(\omega_m)\big), & 1 \le m \le M; \\ \kappa^{\star}_{\mathbb{C}}\big(\boldsymbol{u}(n), \boldsymbol{u}(\omega_m)\big), & M+1 \le m \le 2M. \end{cases}$$

Let us define a new vector that separates the real and imaginary parts of $\boldsymbol{u}(n)$ such that $\tilde{\boldsymbol{u}}(n) = \text{col}\{\boldsymbol{u}_I(n), \boldsymbol{u}_Q(n)\} \in \mathbb{R}^{2L}$. With the Gaussian kernels (5.8)–(5.9), the expected value of (5.37) can be obtained by making use of the moment generating function. We can obtain

$$[\boldsymbol{R}_{\kappa\kappa}]_{i,j} = \big|\boldsymbol{I} + \frac{2}{\xi^2}\boldsymbol{H}(i,j)\boldsymbol{R}_{\tilde{u}}\big|^{-\frac{1}{2}} \cdot \exp\left(-\frac{1}{2\xi^2}[\textstyle\sum_{s=\{i,j\}}\|\boldsymbol{u}_I(\omega_s)\|^2 - \sum_{s=\{i,j\}}\|\boldsymbol{u}_Q(\omega_s)\|^2]\right)$$
$$\times \exp\left(\frac{1\text{i}}{\xi^2}[\delta_i\, \boldsymbol{u}_I^{\top}(\omega_i)\boldsymbol{u}_Q(\omega_i) - \delta_j\, \boldsymbol{u}_I^{\top}(\omega_j)\boldsymbol{u}_Q(\omega_j)]\right) \cdot \exp\left(\frac{1}{2\xi^4}\,\boldsymbol{b}^{\top}\boldsymbol{R}_{\tilde{u}}(\boldsymbol{I} + \frac{2}{\xi^2}\boldsymbol{H}(i,j)\boldsymbol{R}_{\tilde{u}})^{-1}\,\boldsymbol{b}\right) \tag{5.38}$$

where δ_m is the indicator function

$$\delta_m = \begin{cases} 1, & 1 \le m \le M \\ -1, & M+1 \le m \le 2M \end{cases} \tag{5.39}$$

and $\boldsymbol{R}_{\tilde{u}} = E\{\tilde{\boldsymbol{u}}(n)\,\tilde{\boldsymbol{u}}^{\top}(n)\}$. The definition of $\boldsymbol{H}(i,j)$ in (5.38) depends on the index i and j as follows:

$$\boldsymbol{H}(i,j) = \begin{pmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & -\boldsymbol{I} \end{pmatrix},\ 1 \le i,j \le M \text{ and } M+1 \le i,j \le 2M$$

$$\boldsymbol{H}(i,j) = \begin{pmatrix} \boldsymbol{I} & 1\mathrm{i}\,\boldsymbol{I} \\ 1\mathrm{i}\,\boldsymbol{I} & -\boldsymbol{I} \end{pmatrix},\ 1 \le i \le M \text{ and } M+1 \le j \le 2M$$

$$\boldsymbol{H}(i,j) = \begin{pmatrix} \boldsymbol{I} & -1\mathrm{i}\,\boldsymbol{I} \\ -1\mathrm{i}\,\boldsymbol{I} & -\boldsymbol{I} \end{pmatrix},\ 1 \le j \le M \text{ and } M+1 \le i \le 2M$$

Vector $\boldsymbol{b}$ in (5.38) is given by

$$\boldsymbol{b} = \begin{pmatrix} -\sum\limits_{s=\{i,j\}} \boldsymbol{u}_I(\omega_s) + 1\mathrm{i}\,[\delta_i\,\boldsymbol{u}_Q(\omega_i) - \delta_j\,\boldsymbol{u}_Q(\omega_j)] \\ -\sum\limits_{s=\{i,j\}} \boldsymbol{u}_Q(\omega_s) + 1\mathrm{i}\,[-\delta_i\,\boldsymbol{u}_I(\omega_i) + \delta_j\,\boldsymbol{u}_I(\omega_j)] \end{pmatrix}. \tag{5.40}$$

Equation (5.36) leads to the following theorem:

Theorem 5.4.1 *(Stability in the mean) Assume the CMIA holds. Then, for any initial condition, given a dictionary $\boldsymbol{\omega}$, the Gaussian ACKLMS algorithm (5.31) asymptotically converges in mean if the step size is chosen to satisfy*

$$0 < \eta < 2/\mathrm{eig}_{\max}(\boldsymbol{R}_{\kappa\kappa}) \tag{5.41}$$

where $\mathrm{eig}_{\max}(\cdot)$ denotes the maximum eigenvalue of its matrix argument. The entries of $\boldsymbol{R}_{\kappa\kappa}$ are given by (5.38).

5.4.2 Mean-square error analysis

Using (5.34) and CMIA, MSE is related to the second-order moment of the weight vector by [Parreira 2012]

$$J_{\mathrm{ms}}(n) = J_{\min} + \mathrm{trace}\big\{\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_{\omega}(n)\big\} \tag{5.42}$$

where $\boldsymbol{C}_{\omega}(n) = E\left\{\boldsymbol{v}_{\omega}(n)\boldsymbol{v}_{\omega}^{H}(n)\right\}$ is the autocorrelation matrix of the weight error vector $\boldsymbol{v}_{\omega}(n)$, and $J_{\min}$ is the minimum MSE given by (5.30). The analysis of the MSE behavior (5.42) requires a recursive model for $\boldsymbol{C}_{\omega}(n)$. Post-multiplying (5.35) by its Hermitian conjugate, taking the expected value, and using CMIA, we get the following recursion for sufficiently small step sizes

$$\boldsymbol{C}_{\omega}(n+1) \approx \boldsymbol{C}_{\omega}(n) - \eta\,[\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_{\omega}(n) + \boldsymbol{C}_{\omega}(n)\boldsymbol{R}_{\kappa\kappa}] + \eta^2\,\boldsymbol{T}_{\omega}(n) + \eta^2\,\boldsymbol{R}_{\kappa\kappa}J_{\min} \tag{5.43}$$

with

$$\boldsymbol{T}_{\omega}(n) = E\big\{\boldsymbol{\kappa}_{\mathbb{H},\omega}(n)\boldsymbol{\kappa}_{\mathbb{H},\omega}^{H}(n)\boldsymbol{v}_{\omega}(n)\boldsymbol{v}_{\omega}^{H}(n)\boldsymbol{\kappa}_{\mathbb{H},\omega}(n)\boldsymbol{\kappa}_{\mathbb{H},\omega}^{H}(n)\big\}. \tag{5.44}$$

Evaluating (5.44) is a significant step in the analysis since $\boldsymbol{\kappa}_{\mathbb{H},\omega}(n)$ is a nonlinear transformation of a quadratic form of $\boldsymbol{u}(n)$. Using CMIA to determine the (i,j)-th element of $\boldsymbol{T}_\omega(n)$ in (5.44) yields

$$[\boldsymbol{T}_\omega(n)]_{i,j} \approx \sum_{\ell=1}^{M}\sum_{p=1}^{M} E\big\{\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n),\boldsymbol{u}(\omega_i)\big)\,\big[\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n),\boldsymbol{u}(\omega_j)\big)\big]^* \times \kappa_{\mathbb{H}}\big(\boldsymbol{u}(n),\boldsymbol{u}(\omega_\ell)\big)\,\big[\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n),\boldsymbol{u}(\omega_p)\big)\big]^*\big\}\cdot[\boldsymbol{C}_\omega(n)]_{\ell,p}. \tag{5.45}$$

This expression can be written as

$$[\boldsymbol{T}_\omega(n)]_{i,j} \approx \operatorname{trace}\big\{\boldsymbol{K}_\omega(i,j)\,\boldsymbol{C}_\omega(n)\big\} \tag{5.46}$$

where the (ℓ,p)-th entry of the matrix $\boldsymbol{K}_\omega(i,j)$ is given by

$$[\boldsymbol{K}_\omega(i,j)]_{\ell,p} = E\big\{\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n),\boldsymbol{u}(\omega_i)\big)[\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n),\boldsymbol{u}(\omega_j)\big)]^*\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n),\boldsymbol{u}(\omega_\ell)\big)[\kappa_{\mathbb{H}}\big(\boldsymbol{u}(n),\boldsymbol{u}(\omega_p)\big)]^*\big\}. \tag{5.47}$$

Similarly, we also rewrite (5.47) in terms of vector $\tilde{\boldsymbol{u}}(n)$ and use the moment generating function. This leads to (5.48) and (5.49):

$$\begin{aligned}[\boldsymbol{K}_\omega(i,j)]_{\ell,p} = {} & \big|\boldsymbol{I}+\frac{2}{\xi^2}\boldsymbol{L}(i,j)\boldsymbol{R}_{\tilde{u}}\big|^{-\frac{1}{2}}\cdot\exp\left(\frac{1}{2\xi^4}\,\boldsymbol{f}^\top\boldsymbol{R}_{\tilde{u}}\big(\boldsymbol{I}+\frac{2}{\xi^2}\boldsymbol{L}(i,j)\boldsymbol{R}_{\tilde{u}}\big)^{-1}\boldsymbol{f}\right)\\ & \times\exp\left(\frac{1\mathrm{i}}{\xi^2}[\delta_i\,\boldsymbol{u}_I^\top(\omega_i)\boldsymbol{u}_Q(\omega_i)-\delta_j\,\boldsymbol{u}_I^\top(\omega_j)\boldsymbol{u}_Q(\omega_j)+\delta_\ell\,\boldsymbol{u}_I^\top(\omega_\ell)\boldsymbol{u}_Q(\omega_\ell)-\delta_p\,\boldsymbol{u}_I^\top(\omega_p)\boldsymbol{u}_Q(\omega_p)]\right)\\ & \times\exp\left(-\frac{1}{2\xi^2}\big(\textstyle\sum_{s=\{i,j,\ell,p\}}\|\boldsymbol{u}_I(\omega_s)\|^2-\sum_{s=\{i,j,\ell,p\}}\|\boldsymbol{u}_Q(\omega_s)\|^2\big)\right)\end{aligned} \tag{5.48}$$

where

$$\boldsymbol{f}=\begin{pmatrix}-\sum_{s=\{i,j,\ell,p\}}\boldsymbol{u}_I(\omega_s)+1\mathrm{i}\,[\delta_i\,\boldsymbol{u}_Q(\omega_i)-\delta_j\,\boldsymbol{u}_Q(\omega_j)+\delta_\ell\,\boldsymbol{u}_Q(\omega_\ell)-\delta_p\,\boldsymbol{u}_Q(\omega_p)]\\ -\sum_{s=\{i,j,\ell,p\}}\boldsymbol{u}_Q(\omega_s)+1\mathrm{i}\,[-\delta_i\,\boldsymbol{u}_I(\omega_i)+\delta_j\,\boldsymbol{u}_I(\omega_j)-\delta_\ell\,\boldsymbol{u}_I(\omega_\ell)+\delta_p\,\boldsymbol{u}_I(\omega_p)]\end{pmatrix}. \tag{5.49}$$

The definition of $\boldsymbol{L}(i,j)$ in (5.48) depends on i and j as follows:

$$\boldsymbol{L}(i,j)=\begin{pmatrix}2\boldsymbol{I} & \boldsymbol{O}\\ \boldsymbol{O} & -2\boldsymbol{I}\end{pmatrix}\begin{cases}1\le i,j,\ell,p\le M\\ 1\le i,j\le M;\ M+1\le \ell,p\le 2M\\ 1\le \ell,p\le M;\ M+1\le i,j\le 2M\\ 1\le i,\ell\le M;\ M+1\le j,p\le 2M\\ 1\le j,p\le M;\ M+1\le i,\ell\le 2M\\ M+1\le i,j,\ell,p\le 2M\end{cases} \tag{5.50a}$$

$$\boldsymbol{L}(i,j)=\begin{pmatrix}2\boldsymbol{I} & 1\mathrm{i}\,\boldsymbol{I}\\ 1\mathrm{i}\,\boldsymbol{I} & -2\boldsymbol{I}\end{pmatrix}\begin{cases}1\le j\le M;\ M+1\le i,\ell,p\le 2M\\ 1\le \ell\le M;\ M+1\le i,j,p\le 2M\\ 1\le j,\ell,p\le M;\ M+1\le i\le 2M\\ 1\le i,j,\ell\le M;\ M+1\le p\le 2M\end{cases} \tag{5.50b}$$

$$\boldsymbol{L}(i,j) = \begin{pmatrix} 2\boldsymbol{I} & -1\mathrm{i}\,\boldsymbol{I} \\ -1\mathrm{i}\,\boldsymbol{I} & -2\boldsymbol{I} \end{pmatrix} \begin{cases} 1 \le i \le M;\ M+1 \le j,\ell,p \le 2M \\ 1 \le p \le M;\ M+1 \le i,j,\ell \le 2M \\ 1 \le i,\ell,p \le M;\ M+1 \le j \le 2M \\ 1 \le i,j,p \le M;\ M+1 \le \ell \le 2M \end{cases} \tag{5.50c}$$

$$\boldsymbol{L}(i,j) = \begin{pmatrix} 2\boldsymbol{I} & 2\mathrm{i}\,\boldsymbol{I} \\ 2\mathrm{i}\,\boldsymbol{I} & -2\boldsymbol{I} \end{pmatrix} 1 \le j,\ell \le M;\ M+1 \le i,p \le 2M \tag{5.50d}$$

$$\boldsymbol{L}(i,j) = \begin{pmatrix} 2\boldsymbol{I} & -2\mathrm{i}\,\boldsymbol{I} \\ -2\mathrm{i}\,\boldsymbol{I} & -2\boldsymbol{I} \end{pmatrix} 1 \le i,p \le M;\ M+1 \le j,\ell \le 2M \tag{5.50e}$$

5.4.3 Steady-state behavior

In order to determine the steady-state of recursion (5.43), we rewrite it in a lexicographic form. Let $\mathrm{vec}\,\{\cdot\}$ denote the operator that stacks the columns of a matrix on top of each other. Vectorizing $\boldsymbol{C}_\omega(n)$ and $\boldsymbol{R}_{\kappa\kappa}$ by $\boldsymbol{c}_\omega(n) = \mathrm{vec}\,\{\boldsymbol{C}_\omega(n)\}$ and $\boldsymbol{r}_{\kappa,\omega} = \mathrm{vec}\,\{\boldsymbol{R}_{\kappa\kappa}\}$, we can rewrite (5.43) as follows

$$\boldsymbol{c}_\omega(n) = \boldsymbol{G}_\omega\,\boldsymbol{c}_\omega(n) + \eta^2\,J_{\min}\,\boldsymbol{r}_{\kappa,\omega} \tag{5.51}$$

with $\boldsymbol{G}_\omega = \boldsymbol{I} - \eta\,(\boldsymbol{G}_{\omega,1} + \boldsymbol{G}_{\omega,2}) + \eta^2\,\boldsymbol{G}_{\omega,3}$. The universal matrix $\boldsymbol{G}_\omega$ is found by the use of the following definitions:

- $\boldsymbol{I}$ is the identity matrix of dimension $4M^2 \times 4M^2$;
- $\boldsymbol{G}_{\omega,1} = \boldsymbol{I} \otimes \boldsymbol{R}_{\kappa\kappa}$, where $\otimes$ denotes the Kronecker product;
- $\boldsymbol{G}_{\omega,2} = \boldsymbol{R}_{\kappa\kappa} \otimes \boldsymbol{I}$;
- $\boldsymbol{G}_{\omega,3}$ is given by $[\boldsymbol{G}_{\omega,3}]_{i+2(j-1)M,\ell+2(p-1)M} = [\boldsymbol{K}_\omega(i,j)]_{\ell,p}$ with $1 \le i,j,\ell,p \le 2M$.

Assuming convergence, the closed-formed solution of the recursion (5.51) in steady-state is given by

$$\boldsymbol{c}_\omega(\infty) = \eta^2\,J_{\min}(\boldsymbol{I} - \boldsymbol{G}_\omega)^{-1}\boldsymbol{r}_{\kappa,\omega}. \tag{5.52}$$

From equation (5.42), the steady-state MSE is finally given by

$$J_{\mathrm{ms}}(\infty) = J_{\min} + \mathrm{trace}\,\{\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_\omega(\infty)\} \tag{5.53}$$

where the second term on the right side is the steady-state EMSE $J_{\mathrm{EMSE},\omega}(\infty) = \mathrm{trace}\,\{\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_\omega(\infty)\}$.

5.5 Simulation results and discussion

This section provides an example of complex nonlinear system identification to check the accuracy of the convergence models of the ACKLMS algorithm. We considered the complex valued input sequence

$$u(n) = \rho_0\,u(n-1) + \sigma_u\,\sqrt{1-\rho_0^2}\,w(n) \tag{5.54}$$

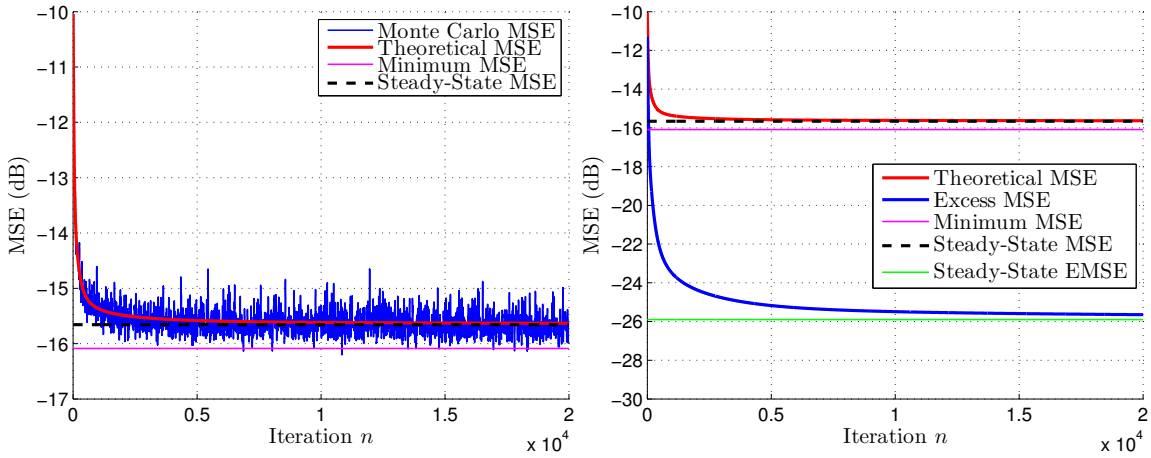
with $w(n) = \sqrt{1-\rho^2}\, w_{\mathrm{re}}(n) + \mathrm{i}\,\rho\, w_{\mathrm{im}}(n)$. Parameter ρ was set to 0.1 corresponding to highly non-circular, and the random variables $w_{\mathrm{re}}(n)$ and $w_{\mathrm{im}}(n)$ were distributed according zero-mean i.i.d. Gaussian distributions with standard deviation $\sigma_w = 1$. Both parameters ρ_0 and σ_u were set to 0.5. The nonlinear complex system to be identified was

$$\begin{cases} y(n) = (0.5 - 0.1\mathrm{i})\, u(n) - (0.3 - 0.2\mathrm{i})\, u(n-1) \\ d(n) = y(n) + (1.25 - 1\mathrm{i})\, y^2(n) + (0.35 - 0.2\mathrm{i})\, y^3(n) + z(n) \end{cases}$$

where $z(n)$ is a complex additive zero-mean Gaussian noise with standard deviation $\sigma_z = 0.1$. At each time n, ACKLMS algorithm was updated with input vector $\boldsymbol{u}(n) = [u(n), u(n-1)]^\top$ and the reference signal $d(n)$. The correlation matrix $\boldsymbol{R}_{\tilde{u}}$ is thus given by

$$\boldsymbol{R}_{\tilde{u}} = \sigma_u^2 \begin{pmatrix} (1-\rho^2) & (1-\rho^2)\rho_0 & 0 & 0 \\ (1-\rho^2)\rho_0 & (1-\rho^2) & 0 & 0 \\ 0 & 0 & \rho^2 & \rho^2\rho_0 \\ 0 & 0 & \rho^2\rho_0 & \rho^2 \end{pmatrix}. \tag{5.55}$$

The pure complex Gaussian bandwidth ξ and the step-size η were set to 0.55 and 0.1, respectively. We used the CS criterion with threshold $\delta_0 = 0.3$ to construct a fixed dictionary



(a) Theory vs. Monte Carlo.

(b) Steady-state results.

Figure 5.4: Simulation results of ACKLMS algorithm ($\rho = 0.1$, $\xi = 0.55$, $\eta = 0.1$ and $M = 12$).

of length $M = 12$. All simulation curves were obtained by averaging over 200 Monte Carlo runs. It is shown in Figure 5.4 that the theoretical curves consistently agree with the Monte Carlo simulations in both transient and steady-state.

5.6 Conclusion

Complex kernel-based adaptive algorithms have been recently introduced for complex-valued nonlinear system identification. These algorithms are built upon the same framework as complex linear adaptive filtering techniques and Wirtinger's calculus in complex

reproducing kernel Hilbert spaces. This chapter introduced two types of complex monokernel LMS algorithm and the ACKLMS algorithm based on the framework of complex multi-kernel. Then we derived a theoretical model of convergence for the ACKLMS algorithm with pre-tuned dictionary. The simulation results demonstrated the effectiveness and accuracy of our model.

Chapter 6

Diffusion adaptation over networks with KLMS

Contents

6.1 Introduction

Distributed learning over networks allows a set of interconnected agents to perform pre-assigned tasks such as detection and estimation from streaming data. Potential applications include, for instance, natural phenomena and infrastructure monitoring, etc. Due to energy constraints, limited communication capabilities and large scale networks, signal processing strategies have moved from centralized solutions with a fusion center [Nguyen 2005] to decentralized cooperative solutions with in-network sensor data processing. For online parameter estimation, a variety of distributed strategies have been proposed. These include incremental strategies [Lopes 2007b], consensus strategies [Kar 2009] and diffusion strategies [Lopes 2007a]. Although the energy cost of communications tends to be minimum, incremental strategies are problematic for large scale applications as they operate on a cyclic path that runs across the nodes, which makes them sensitive to link failures. With diffusion modes of cooperation, the agents cooperate with each other through local interactions that consist of exchanging raw data and local estimates. Both of the mentioned cooperation modes over networks are depicted in Figure 6.1. Diffusion strategies are attractive because they are scalable, robust, and enable continuous adaptation and learning. Moreover, they have shown to have superior

stability properties and performance [Sayed 2013b, Sayed 2013a] than consensus strategies [Xiao 2004, Kar 2009, Srivastava 2011].

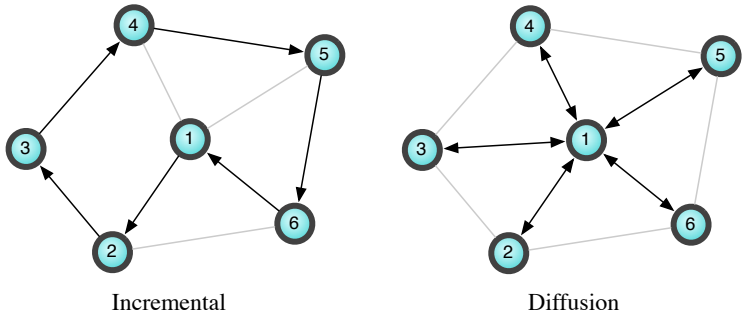


Figure 6.1: Cooperation models over networks.

Decentralized detection and estimation have often been considered with parametric models, in which the statistics of observations are assumed known. Such assumptions are usually motivated by prior application-specific domain knowledge. Robust nonparametric methods are however desirable when few prior information is available. To address such situations, nonparametric methods based on kernel functions were primarily considered for decentralized detection and estimation over networks [Guestrin 2004, Nguyen 2005]. The successive orthogonal projection (SOP) algorithm was derived to address distributed learning problems over networks with kernel-based models [Predd 2005, Predd 2006]. An incremental kernel-based strategy was introduced in [Honeine 2009]. Finally, a linear combination of Gaussian functions was considered in [Bergamo 2012] for estimating scalar fields with diffusion networks.

In this chapter, we introduce functional diffusion strategies in reproducing kernel Hilbert spaces with distributed KLMS algorithm. This chapter is organized as follows. In Section 6.2, we introduce some basic principles on online learning with the KLMS algorithm again. In Section 6.3, a functional framework for diffusion adaption over networks is derived. In Section 6.4, we present some illustrative simulation results.

6.2 The kernel least-mean-square algorithm

In order to introduce the kernel least-mean-square algorithm into distributed networks setting, we firstly recall the derivation of KLMS algorithm in detail. Let $\mathcal{H}$ be a Hilbert space of functions ψ from a subspace $\mathcal{U}$ of $\mathbb{R}^L$ to $\mathbb{R}$. Assume that $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS), that is, there exists a map $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ such that:

$$\forall \boldsymbol{u} \in \mathcal{U}, \quad \kappa(\boldsymbol{u}, \cdot) \in \mathcal{H} \tag{6.1a}$$

$$\forall \psi \in \mathcal{H}, \quad \psi(\boldsymbol{u}) = \langle \psi, \kappa(\boldsymbol{u}, \cdot) \rangle_{\mathcal{H}} \tag{6.1b}$$

Consider the kernel least-squares problem. Given pairs of input vectors and desired outputs $\{(\boldsymbol{u}_i, d(i))\}_i$, which satisfy the model

$$d(i) = \psi_{\text{opt}}(\boldsymbol{u}_i) + v(i) \tag{6.2}$$

where $v(i)$ is a zero-mean white noise with power σ_v^2, the problem is to estimate ψ_{opt} such that:

$$\psi_{\text{opt}} = \arg\min_{\psi\in\mathcal{H}} J(\psi) \quad \text{with} \quad J(\psi) = E\{|d(i) - \psi(\boldsymbol{u}_i)|^2\}. \tag{6.3}$$

Calculating the Fréchet derivative of $J(\psi)$ with respect to ψ we find:

$$\nabla J(\psi) = -2\, E\{[d(i) - \psi(\boldsymbol{u}_i)]\, \kappa(\cdot, \boldsymbol{u}_i)\}. \tag{6.4}$$

The desired function ψ_{opt} satisfies the normal equation:

$$E\{\psi_{\text{opt}}(\boldsymbol{u}_i)\, \kappa(\cdot, \boldsymbol{u}_i)\} = E\{d(i)\, \kappa(\cdot, \boldsymbol{u}_i)\}. \tag{6.5}$$

It is seen from (6.3) that:

$$\begin{aligned} J(\psi) &= \sigma_d^2 - 2\langle E\{d(i)\, \kappa(\cdot, \boldsymbol{u}_i)\}, \psi\rangle_{\mathcal{H}} + E\{\psi^2(\boldsymbol{u}_i)\} \\ &= \sigma_d^2 - 2\langle E\{\psi_{\text{opt}}(\boldsymbol{u}_i)\, \kappa(\cdot, \boldsymbol{u}_i)\}, \psi\rangle_{\mathcal{H}} + E\{\psi^2(\boldsymbol{u}_i)\} \\ &= \sigma_d^2 - 2E\{\langle \psi_{\text{opt}}(\boldsymbol{u}_i), \psi(\boldsymbol{u}_i)\rangle_{\mathcal{H}}\} + E\{\psi^2(\boldsymbol{u}_i)\} \end{aligned} \tag{6.6}$$

where σ_d^2 denotes $E\{d^2(i)\}$. The first and the third equalities follow from (6.1b). The second equality follows from (6.5). We see that $J(\psi_{\text{opt}}) = \sigma_d^2 - E\{[\psi_{\text{opt}}(\boldsymbol{u}_i)]^2\}$. A completion-of-squares argument finally shows that $J(\psi)$ can be expressed as

$$J(\psi) = J(\psi_{\text{opt}}) + E\{[\psi(\boldsymbol{u}_i) - \psi_{\text{opt}}(\boldsymbol{u}_i)]^2\}. \tag{6.7}$$

We shall use this expression in the sequel, where $J(\psi_{\text{opt}})$ and $J_{\min}$ will interchangeably denote the minimum cost value of $J(\psi)$.

The optimal implementation (6.5) for determining ψ_{opt} requires knowledge of the data moments. This information is usually unavailable. Stochastic-gradient methods are popular adaptive learning algorithms obtained from gradient-descent implementations by replacing the required derivatives by some suitable approximations. One of the simplest approximations for $\nabla J(\psi)$ consists of replacing the random variables in (6.3) by the observations at iteration i, namely,

$$-\nabla J(\psi) \approx [d(i) - \psi(\boldsymbol{u}_i)]\, \kappa(\cdot, \boldsymbol{u}_i). \tag{6.8}$$

The corresponding steepest-descent recursion is widely known as the kernel least-mean-squares (KLMS) algorithm [Richard 2009]:

$$\psi_i = \psi_{i-1} + \mu\, [d(i) - \psi_{i-1}(\boldsymbol{u}_i)]\, \kappa(\cdot, \boldsymbol{u}_i) \tag{6.9}$$

where μ is a small positive step-size. Despite its computational simplicity, the main drawback of KLMS is that an increasing number of kernel functions $\kappa(\cdot, \boldsymbol{u}_i)$ is involved in the estimation process as new data $\boldsymbol{u}_i$ are collected. To overcome this limitation, finite-size models of the form

$$\psi = \sum_{j=1}^{M} \alpha_j\, \kappa(\,\cdot\,, \boldsymbol{u}_{\omega_j}) \tag{6.10}$$

and sparsity-promoting strategies are usually considered in the literature [Richard 2009], where $\mathcal{D} = \{\kappa(\cdot, \boldsymbol{u}_{\omega_j})\}_j$ is a dictionary learnt from the input data $\{\kappa(\cdot, \boldsymbol{u}_i)\}_i$. Then, KLMS

reduces to a two-alternative choice procedure at each instant i: (Acceptance case) a dictionary learning stage that inserts $\kappa(\cdot, \boldsymbol{u}_i)$ into $\mathcal{D}_{i-1}$ if some given sparsification rule, such as the coherence rule below, is satisfied; (Rejection case) Otherwise, an adaptation step to update the vector $\boldsymbol{\alpha}$ of parameters α_j.

- **Rejection case:** $\max_{j=1,\ldots,\text{card}(\mathcal{D}_{i-1})} |\kappa(\boldsymbol{u}_i, \boldsymbol{u}_{\omega_j})| \leq \delta_0$

$$\begin{aligned} \boldsymbol{\alpha}_i &= \begin{bmatrix} \boldsymbol{\alpha}_{i-1} \\ 0 \end{bmatrix} + \mu\, e_i\, \boldsymbol{\kappa}_i \\ \mathcal{D}_i &= \mathcal{D}_{i-1} \cup \{\kappa(\cdot, \boldsymbol{u}_i)\} \end{aligned} \tag{6.11}$$

- **Acceptance case:** $\max_{j=1,\ldots,\text{card}(\mathcal{D}_{i-1})} |\kappa(\boldsymbol{u}_i, \boldsymbol{u}_{\omega_j})| > \delta_0$

$$\begin{aligned} \boldsymbol{\alpha}_i &= \boldsymbol{\alpha}_{i-1} + \mu\, e_i\, \boldsymbol{\kappa}_i \\ \mathcal{D}_i &= \mathcal{D}_{i-1} \end{aligned} \tag{6.12}$$

where $\boldsymbol{\kappa}_i = [\kappa(\boldsymbol{u}_i, \boldsymbol{u}_{\omega_1}), \ldots, \kappa(\boldsymbol{u}_i, \boldsymbol{u}_{\omega_{\text{card}(\mathcal{D}_{i-1})}})]^\top$ is the kernelized input, and δ_0 is the threshold of CS criterion in $[0, 1)$. In [Richard 2009], it is shown that the dictionary learning step converges to a dictionary $\mathcal{D}$ of finite size, say M, and the algorithm above reduces to (6.12) after a finite number of iterations.

6.3 Diffusion adaptation with KLMS algorithm

Consider a collection of N agents interested in estimating the same function ψ_{opt} of $\mathcal{H}$ from data realizations $\big(\boldsymbol{u}_{k,i}, d_k(i)\big)$, which satisfy a model of the form

$$d_k(i) = \psi_{\text{opt}}(\boldsymbol{u}_{k,i}) + v_k(i) \tag{6.13}$$

where $v_k(i)$ is a zero-mean white noise with power $\sigma_{v,k}^2$. To recover this unknown function ψ_{opt}, our strategy is to optimize the following global cost function in a distributed manner:

$$J(\psi) = \sum_{k=1}^{N} E\{|d_k(i) - \psi(\boldsymbol{u}_{k,i})|^2\}. \tag{6.14}$$

Assume that the set of neighbors connected with the ℓ-th agent is fixed and denoted by $\mathcal{N}_\ell$. We can express $J(\psi)$ as follows:

$$J(\psi) = \sum_{\ell=1}^{N} J_\ell^{\text{loc}}(\psi) \tag{6.15}$$

with

$$J_\ell^{\text{loc}}(\psi) = \sum_{k \in \mathcal{N}_\ell} c_{k\ell}\, E\{|d_k(i) - \psi(\boldsymbol{u}_{k,i})|^2\} \tag{6.16}$$

where $\{c_{k\ell}\}$ is a set of nonnegative coefficients, freely chosen by the designer, that satisfy:

$$c_{k\ell} = 0 \;\text{ if }\; k \notin \mathcal{N}_\ell \quad \text{and} \quad \sum_{\ell=1}^{N} c_{k\ell} = 1. \tag{6.17}$$

We collect the coefficients $\{c_{k\ell}\}$ into an $N \times N$ matrix $\boldsymbol{C}$, which is right stochastic since each row of $\boldsymbol{C}$ adds up to one.

Consider the local cost function $J_\ell^{\text{loc}}(\psi)$ at each node ℓ. It follows from (6.7) that:

$$J_\ell^{\text{loc}}(\psi) = J_{\ell,\min}^{\text{loc}} + \sum_{k\in\mathcal{N}_\ell} c_{k\ell}\, E\{|\psi(\boldsymbol{u}_{k,i}) - \psi_{\text{opt}}(\boldsymbol{u}_{k,i})|^2\} \tag{6.18}$$

where $J_{\ell,\min}^{\text{loc}} = J_\ell^{\text{loc}}(\psi_{\text{opt}})$. Substituting (6.18) into (6.15), and dropping the term that does not depend on ψ, we obtain the following alternative global cost function:

$$J(\psi) = J_n^{\text{loc}}(\psi) + \sum_{\ell\neq n}\sum_{k\in\mathcal{N}_\ell} c_{k\ell}\, E\{|\psi(\boldsymbol{u}_{k,i}) - \psi_{\text{opt}}(\boldsymbol{u}_{k,i})|^2\}. \tag{6.19}$$

In this expression, the minimizer ψ_{opt} in the correction term that relates the global cost function to the local cost function at every node n is not known since the nodes wish to estimate it. This issue is addressed in the sequel. Likewise, not all information needed to compute the expected value are available to node n since it can only have access to information from its neighbors. We thus introduce the modified local cost function at node n:

$$J_n(\psi) = J_n^{\text{loc}}(\psi) + \sum_{\ell\in\mathcal{N}_n\setminus\{n\}}\sum_{k\in\mathcal{N}_\ell} c_{k\ell}\, E\{|\psi(\boldsymbol{u}_{k,i}) - \psi_{\text{opt}}(\boldsymbol{u}_{k,i})|^2\}. \tag{6.20}$$

The probability density functions required to calculate the expected values may not be available because often nodes can only observe realizations $\boldsymbol{u}_{k,i}$. To address this issue, note that:

$$\begin{aligned} E\{|\psi(\boldsymbol{u}_{k,i}) - \psi_{\text{opt}}(\boldsymbol{u}_{k,i})|^2\} &= \int |\psi(\boldsymbol{u}_{k,i}) - \psi_{\text{opt}}(\boldsymbol{u}_{k,i})|^2\, dP(\boldsymbol{u}_{k,i}) \\ &\leq \|\psi - \psi_{\text{opt}}\|_{\mathcal{H}}^2 \int \kappa(\boldsymbol{u}_{k,i}, \boldsymbol{u}_{k,i})\, dP(\boldsymbol{u}_{k,i}) \\ &\leq M\|\psi - \psi_{\text{opt}}\|_{\mathcal{H}}^2 \end{aligned} \tag{6.21}$$

where P is a probability measure. The first and the second inequality follow from the Cauchy-Schwarz inequality and the boundedness of the kernel κ, respectively. We suggest to replace the second term on the RHS of (6.20) by the following upper-bound:

$$\sum_{k\in\mathcal{N}_\ell} c_{k\ell}\, E\{|\psi(\boldsymbol{u}_{k,i}) - \psi_{\text{opt}}(\boldsymbol{u}_{k,i})|^2\} \leq b_{\ell n}\|\psi - \psi_{\text{opt}}\|_{\mathcal{H}}^2 \tag{6.22}$$

where $b_{\ell n}$ is some nonnegative coefficient. The modified cost function (6.20) is then relaxed as follows:

$$J_n'(\psi) = J_n^{\text{loc}}(\psi) + \sum_{\ell\in\mathcal{N}_n\setminus\{n\}} b_{\ell n}\|\psi - \psi_{\text{opt}}\|_{\mathcal{H}}^2. \tag{6.23}$$

With the exception of ψ_{opt}, the cost (6.23) at node n relies solely on information available to this node from its neighborhood.

Node n can compute successive steepest-descent iterations to minimize $J_n'(\psi)$. Let $\psi_{n,i-1}$ be the estimate for ψ_{opt} by node n at time $i-1$. The update from $\psi_{n,i-1}$ to $\psi_{n,i}$ can be performed as:

$$\psi_{n,i} = \psi_{n,i-1} - \mu_n \nabla J_n'(\psi_{n,i-1}), \quad \psi_{n,-1} = \text{initial guess} \tag{6.24}$$

where μ_n is a small positive step-size at node n. Computing the Fréchet derivative of (6.23), and dropping the expectation operator from the definition of $J_n^{\text{loc}}(\psi)$ to use instantaneous approximations instead, we get:

$$\begin{aligned}\psi_{n,i} = \psi_{n,i-1} &+ \mu_n \sum_{\ell \in \mathcal{N}_n} c_{\ell n} \left[d_\ell(i) - \psi_{n,i-1}(\boldsymbol{u}_{\ell,i}) \right] \kappa(\cdot, \boldsymbol{u}_{\ell,i}) \\ &+ \mu_n \sum_{\ell \in \mathcal{N}_n \setminus \{n\}} b_{\ell n} (\psi_{\text{opt}} - \psi_{\ell,i-1}).\end{aligned} \tag{6.25}$$

Among other possible forms, we can implement (6.25) in two successive steps involving each one a correction term as follows:

$$\varphi_{n,i} = \psi_{n,i-1} + \mu_n \sum_{\ell \in \mathcal{N}_n} c_{\ell n} \left[d_\ell(i) - \psi_{n,i-1}(\boldsymbol{u}_{\ell,i}) \right] \kappa(\cdot, \boldsymbol{u}_{\ell,i-1}) \tag{6.26a}$$

$$\psi_{n,i} = \varphi_{n,i} + \mu_n \sum_{\ell \in \mathcal{N}_n \setminus \{n\}} b_{\ell n} (\psi_{\text{opt}} - \psi_{n,i-1}) \tag{6.26b}$$

First, in (6.26b), neither node n nor its neighbors know the optimum function ψ_{opt}. Each node ℓ can however use its local intermediate estimate $\varphi_{\ell,i}$ as an approximation. Second, $\psi_{n,i-1}$ in (6.26b) can be advantageously replaced by $\varphi_{n,i}$ since it is obtained by incorporating information from the neighbors in (6.26a). Step (6.26b) then becomes:

$$\psi_{n,i} = \left(1 - \mu_n \sum_{\ell \in \mathcal{N}_n \setminus \{n\}} b_{\ell n} \right) \varphi_{n,i} + \mu_n \sum_{\ell \in \mathcal{N}_n} b_{\ell n} \, \varphi_{\ell,i}. \tag{6.27}$$

We introduce the following weighting coefficients:

$$\begin{aligned} a_{nn} &= 1 - \mu_n \sum_{\ell \in \mathcal{N}_n \setminus \{n\}} b_{\ell n} \\ a_{\ell n} &= \mu_n \, b_{\ell n}, \quad \ell \in \mathcal{N}_n \setminus \{n\} \\ a_{\ell n} &= 0, \quad \ell \notin \mathcal{N}_n \end{aligned} \tag{6.28}$$

and collect these coefficients into a $(N \times N)$ matrix $\boldsymbol{A}$. For sufficiently small step-sizes μ_n, observe that the coefficients $\{a_{\ell n}\}$ are nonnegative and each column of $\boldsymbol{A}$ adds up to one. Just like the coefficients $\{c_{\ell n}\}$, the coefficients $\{a_{\ell n}\}$ can be freely chosen by the designer provided that $\boldsymbol{A}$ is left stochastic.

6.3.1 Functional adapt-then-Combine diffusion strategy

Substituting the so-called coefficients $\{a_{\ell n}\}$ into (6.27), we arrive at the following diffusion strategy:

$$\boxed{\begin{aligned} \varphi_{n,i} &= \psi_{n,i-1} + \mu_n \sum_{\ell \in \mathcal{N}_n} c_{\ell n} \left[d_\ell(i) - \psi_{n,i-1}(\boldsymbol{u}_{\ell,i}) \right] \kappa(\cdot, \boldsymbol{u}_{\ell,i}) \\ \psi_{n,i} &= \sum_{\ell \in \mathcal{N}_n} a_{\ell n} \, \varphi_{\ell,i} \end{aligned}} \tag{6.29}$$

These are the functional adapt-then-combine (FATC) diffusion algorithm, whose scheme is plotted in the Figure 6.2.

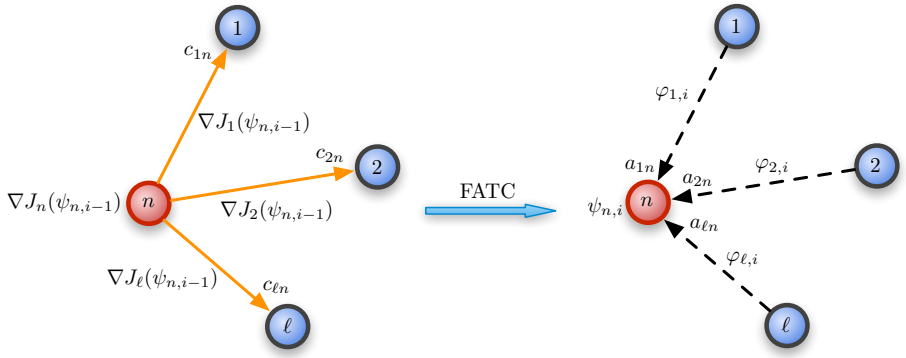


Figure 6.2: The scheme of FATC diffusion algorithm.

6.3.2 Functional Combine-then-adapt diffusion strategy

Similarly, returning to (6.25) and considering the second correction first, we get the alternative diffusion strategy:

$$\boxed{\begin{aligned}\varphi_{n,i-1} &= \sum_{\ell\in\mathcal{N}_n} a_{\ell n}\,\psi_{\ell,i-1}\\ \psi_{n,i} &= \varphi_{n,i-1} + \mu_n \sum_{\ell\in\mathcal{N}_n} c_{\ell n}\,[d_\ell(i) - \varphi_{n,i-1}(\boldsymbol{u}_{\ell,i})]\,\kappa(\cdot,\boldsymbol{u}_{\ell,i})\end{aligned}} \tag{6.30}$$

Correspondingly, the above strategy is called the functional combine-then-adapt (FCTA) diffusion. The scheme of FCTA diffusion algorithm is given in the Figure 6.3.

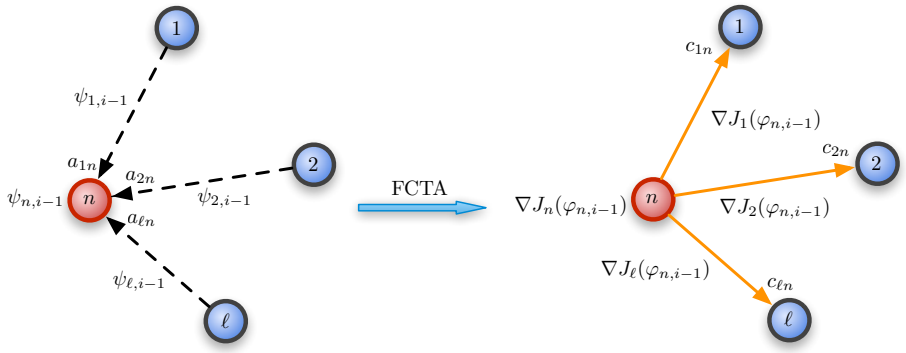


Figure 6.3: The scheme of FCTA diffusion algorithm.

6.3.3 Implementation

Online processing of time series data raises the question of how to process an increasing amount of observations $\boldsymbol{u}_{\ell,i}$ as new data is collected at each node. Indeed, as the KLMS algorithm (6.9), an undesirable characteristic of FATC and FCTA algorithms (6.29)–(6.30)

is that the number of terms in $\varphi_{n,i}$ and $\psi_{n,i}$ grows linearly with the number of input data. This dramatically increases the computational burden and memory requirement. To overcome this barrier, in this chapter, we shall consider as a prior that nodes share a dictionary $\mathcal{D}$ of finite size M. We leave this sharing processing, which should be based on the coherence rule, for future work.

Then, we can write the estimated inference function of $\varphi_{n,i}$ and $\psi_{n,i}$ in (6.29)–(6.30) as:

$$\begin{cases} \varphi_{n,i} &= \boldsymbol{\beta}_{n,i}^{\top} \boldsymbol{\kappa}_{n,i} \\ \psi_{n,i} &= \boldsymbol{\alpha}_{n,i}^{\top} \boldsymbol{\kappa}_{n,i} \end{cases} \tag{6.31}$$

with the kernelized input vector $\boldsymbol{\kappa}_{n,i} = [\kappa(\boldsymbol{u}_{n,i}, \boldsymbol{u}_{\omega_1}), \ldots, \kappa(\boldsymbol{u}_{n,i}, \boldsymbol{u}_{\omega_M})]^{\top}$, the weight vector $\boldsymbol{\alpha}_{n,i} = [\alpha_{n,i}(1), \ldots, \alpha_{n,i}(M)]^{\top}$ and the intermediate weight vector $\boldsymbol{\beta}_{n,i} = [\beta_{n,i}(1), \ldots, \beta_{n,i}(M)]^{\top}$. As a consequence, the FATC strategy can be expressed as follows:

$$\boxed{\begin{aligned} \boldsymbol{\beta}_{n,i} &= \boldsymbol{\alpha}_{n,i-1} + \mu_n \sum_{\ell \in \mathcal{N}_n} c_{\ell n} \left[d_{\ell}(i) - \boldsymbol{\alpha}_{n,i-1}^{\top} \boldsymbol{\kappa}_{\ell,i} \right] \boldsymbol{\kappa}_{\ell,i} \\ \boldsymbol{\alpha}_{n,i} &= \sum_{\ell \in \mathcal{N}_n} a_{\ell n} \boldsymbol{\beta}_{\ell,i} \end{aligned}} \tag{6.32}$$

We obtain the KLMS-FATC algorithm. Similarly, the KLMS algorithm with FCTA strategy is given by:

$$\boxed{\begin{aligned} \boldsymbol{\beta}_{n,i-1} &= \sum_{\ell \in \mathcal{N}_n} a_{\ell n} \boldsymbol{\alpha}_{\ell,i-1} \\ \boldsymbol{\alpha}_{n,i} &= \boldsymbol{\beta}_{n,i-1} + \mu_n \sum_{\ell \in \mathcal{N}_n} c_{\ell n} \left[d_{\ell}(i) - \boldsymbol{\beta}_{n,i-1}^{\top} \boldsymbol{\kappa}_{\ell,i} \right] \boldsymbol{\kappa}_{\ell,i} \end{aligned}} \tag{6.33}$$

6.3.4 Stability of functional diffusion strategy in the mean

In order to make the study of the weight estimate of functional diffusion strategy in mean sense trackable, assume a simple case that all the nodes use an identical M-dimension dictionary $\mathcal{D} = [\boldsymbol{u}_{\omega_1}, \ldots, \boldsymbol{u}_{\omega_M}]^{\top}$ preassigned a prior.

Without loss of generality, we only consider the parametric support vector diffusion structure of FATC strategy (6.32). The non-negative real coefficients $a_{\ell n}, c_{\ell n}$ are the (ℓ, n)-th entries of a left-stochastic matrices $\boldsymbol{A}$ and right-stochastic $\boldsymbol{C}$, that is,

$$\boldsymbol{A}^{\top} \mathbf{1}_N = \mathbf{1}_N, \quad \boldsymbol{C}^{\top} \mathbf{1}_N = \mathbf{1}_N \tag{6.34}$$

and satisfy

$$a_{1,\ell n} = 0, \quad c_{\ell n} = 0 \quad \text{if} \quad \ell \notin \mathcal{N}_n. \tag{6.35}$$

Let us define the network MSE by averaging errors over N nodes:

$$\frac{1}{N} \sum_{k=1}^{N} E\{|d_k(i) - \psi(\boldsymbol{u}_{k,i})|^2\}. \tag{6.36}$$

Let us introduce $N\times 1$ block optimum weight vector, block weight estimate and intermediate weight estimate vectors, whose individual entries are of size $M \times 1$ each:

$$\boldsymbol{\alpha}_{\text{opt}} = \text{col}\{\boldsymbol{\alpha}_{\text{opt},1}, \ldots, \boldsymbol{\alpha}_{\text{opt},N}\}; \tag{6.37}$$

$$\boldsymbol{\alpha}_{n} = \text{col}\{\boldsymbol{\alpha}_{1,i}, \ldots, \boldsymbol{\alpha}_{N,i}\}; \tag{6.38}$$

$$\boldsymbol{\beta}_{n} = \text{col}\{\boldsymbol{\beta}_{1,i}, \ldots, \boldsymbol{\beta}_{N,i}\}. \tag{6.39}$$

The weight error vector for each node n at iteration i is defined by $\boldsymbol{v}_{n,i} = \boldsymbol{\alpha}_{n,i} - \boldsymbol{\alpha}_{\text{opt}}$. The weight error vector $\boldsymbol{v}_{n,i}$ are also stacked on top of each other to get the block weight error vector defined as follow:

$$\boldsymbol{v}_{i} = \text{col}\{\boldsymbol{v}_{1,i}, \ldots, \boldsymbol{v}_{N,i}\}. \tag{6.40}$$

Assumption 2 *$\boldsymbol{\kappa}_{\ell,i}\boldsymbol{\kappa}_{\ell,i}^{\top}$ is independent of $\boldsymbol{v}_{n,i}$.*
This assumption, called modified independence assumption (MIA), is justified in detail in [Minkoff 2001]. It has been successfully employed in several adaptive filter analyses, and has been shown in [Minkoff 2001] to be less restrictive than the well known independence assumption (IA) [Sayed 2008]. It is called here for further reference *conditioned* MIA, or CMIA, to distinguish it from the MIA used in [Parreira 2012].

The estimation error in the second step (6.32) of diffusion KLMS can be rewritten as

$$d_{\ell}(i) - \boldsymbol{\alpha}_{n,i-1}^{\top}\boldsymbol{\kappa}_{\ell,i} = d_{\ell}(i) - \boldsymbol{\kappa}_{\ell,i}^{\top}\boldsymbol{v}_{n,i} - \boldsymbol{\kappa}_{\ell,i}^{\top}\boldsymbol{\alpha}_{\text{opt}}. \tag{6.41}$$

Subtracting optimum weight vector $\boldsymbol{\alpha}_{\text{opt},n}$ from both sides of the equation (6.32), and using the above expression, yields

$$\boldsymbol{\beta}_{n,i-1} - \boldsymbol{\alpha}_{\text{opt},n} = \big(\boldsymbol{I}_M - \mu_n \sum_{\ell\in\mathcal{N}_n} c_{\ell n}\, \boldsymbol{\kappa}_{\ell,i}\boldsymbol{\kappa}_{\ell,i}^{\top}\big)\boldsymbol{v}_{n,i-1} + \mu_n \sum_{\ell\in\mathcal{N}_n} c_{\ell n}\, [d_{\ell}(i) - \boldsymbol{\kappa}_{\ell,i}^{\top}\boldsymbol{\alpha}_{\text{opt}}]\boldsymbol{\kappa}_{\ell,i} \tag{6.42}$$

which can be expressed in block-based form:

$$\boldsymbol{\beta}_{i-1} - \boldsymbol{\alpha}_{\text{opt}} = \big(\boldsymbol{I}_{MN} - \boldsymbol{\mathcal{M}}\boldsymbol{H}_i\big)\boldsymbol{v}_{i-1} + \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^{\top}\boldsymbol{s}_{i-1} \tag{6.43}$$

with $\boldsymbol{\mathcal{C}} = \boldsymbol{C}\otimes\boldsymbol{I}_M$. In (6.43) the $(N\times N)$ block diagonal matrices, whose individual entries are of size $M\times M$, are defined as

$$\boldsymbol{\mathcal{M}} = \text{Diag}\big\{\mu_1\boldsymbol{I}_M, \ldots, \mu_N\boldsymbol{I}_M\big\}, \tag{6.44}$$

$$\boldsymbol{H}_i = \text{Diag}\big\{\sum_{\ell\in\mathcal{N}_1} c_{\ell 1}\, \boldsymbol{\kappa}_{\ell,i}\boldsymbol{\kappa}_{\ell,i}^{\top}, \ldots, \sum_{\ell\in\mathcal{N}_N} c_{\ell N}\, \boldsymbol{\kappa}_{\ell,i}\boldsymbol{\kappa}_{\ell,i}^{\top}\big\} \tag{6.45}$$

and the $NM\times 1$ block column vector is defined as

$$\boldsymbol{s}_{i-1} = \text{col}\big\{[d_n(i) - \boldsymbol{\kappa}_{n,i}^{\top}\boldsymbol{\alpha}_{\text{opt}}]\boldsymbol{\kappa}_{n,i}\big\}_{n=1}^{N}. \tag{6.46}$$

Defining $\boldsymbol{\mathcal{A}} = \mathcal{A}\otimes\boldsymbol{I}_M$, the combination step (6.32) leads to

$$\boldsymbol{\alpha}_{i} = \boldsymbol{\mathcal{A}}^{\top}\boldsymbol{\beta}_{i}. \tag{6.47}$$

Subtracting $\boldsymbol{\alpha}_{\text{opt}}$ from both sides of (6.47) and using (6.43), the stochastic recursion for $\boldsymbol{v}_i$ can be written as compact form

$$\boldsymbol{v}_i = \boldsymbol{\mathcal{A}}^\top \big(\boldsymbol{I}_{MN} - \boldsymbol{\mathcal{M}}\boldsymbol{H}_i\big)\boldsymbol{v}_{i-1} + \boldsymbol{\mathcal{A}}^\top \boldsymbol{\mathcal{M}}\boldsymbol{\mathcal{C}}^\top \boldsymbol{s}_{i-1}. \tag{6.48}$$

We assume that the finite-order model provides a close enough approximation to the infinite-order model with minimum MSE, so that $E\{d_\ell(i) - \boldsymbol{\kappa}_{\ell,i}^\top \boldsymbol{\alpha}_{\text{opt}}\} = 0$. Taking the expectation of both sides, and using CMIA assumption, we can obtain the mean weight error model

$$E\{\boldsymbol{v}_i\} = \boldsymbol{\mathcal{A}}^\top \big(\boldsymbol{I}_{MN} - \boldsymbol{\mathcal{M}}\boldsymbol{H}\big)E\{\boldsymbol{v}_{i-1}\}, \quad i \geq 0 \tag{6.49}$$

where $\boldsymbol{H}$ is the expected matrix of $\boldsymbol{H}_i$, namely

$$\boldsymbol{H} = E\{\boldsymbol{H}_i\} = \text{Diag}\{\boldsymbol{R}_1, \ldots, \boldsymbol{R}_N\} \tag{6.50}$$

with $\boldsymbol{R}_n = \sum_{\ell \in \mathcal{N}_n} c_{\ell n}\, \boldsymbol{R}_{\kappa\kappa,\ell}$. Note that $\boldsymbol{R}_{\kappa\kappa,n}$ is the correlation matrix of kernelized input, whose each entries were given in [Chen 2014a]. Likewise, the mean weight error model of support parameter-vector formulation of FCTA strategy can be also obtained

$$E\{\boldsymbol{v}_i\} = \big(\boldsymbol{I}_{MN} - \boldsymbol{\mathcal{M}}\boldsymbol{H}\big)\boldsymbol{\mathcal{A}}^\top E\{\boldsymbol{v}_{i-1}\}, \quad i \geq 0. \tag{6.51}$$

Theorem 6.3.1 *(Stability in the mean) Based on the problem of (6.3), select a right stochastic matrix $\boldsymbol{C}$ and left stochastic matrix $\boldsymbol{A}$. Assume CMIA holds. When the kernel adaptive diffusion algorithms (6.32) and (6.33) across the network asymptotically converge in the mean if the step-size parameters μ_n are chosen to satisfy*

$$\mu_n < \frac{2}{\text{eig}_{\max}(\boldsymbol{R}_n)} \tag{6.52}$$

where $\text{eig}_{\max}(\cdot)$ denotes the maximum eigenvalue of the matrix.

Observe that the the mean stability condition (6.52) does not depend on the specific combination matrix $\boldsymbol{A}$. Only the combination matrix $\boldsymbol{C}$ influences the condition on the step-size through the neighborhood covariance matrices $\{\boldsymbol{R}_n\}$.

6.4 Simulation results and discussion

In this section, we address two identification problems with the FATC and FCTA diffusion strategies over distributed networks. The dictionaries $\mathcal{D}_n$ are fixed. These experiments will illustrate the effectiveness of the proposed algorithms with a fixed dictionary. All the learning curves were obtained by averaging over 200 Monte Carlo runs.

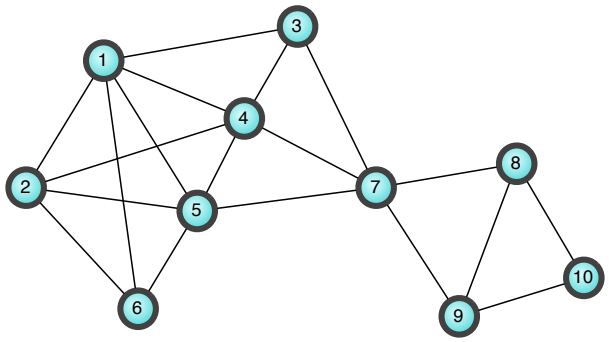
6.4.1 Example 1

We consider $N = 10$ sensors randomly deployed at 101×101 square grid. The topology of the network is depicted in Figure 6.4(a). Each node k observes the desired output $\{d_k(i)\}$ given by

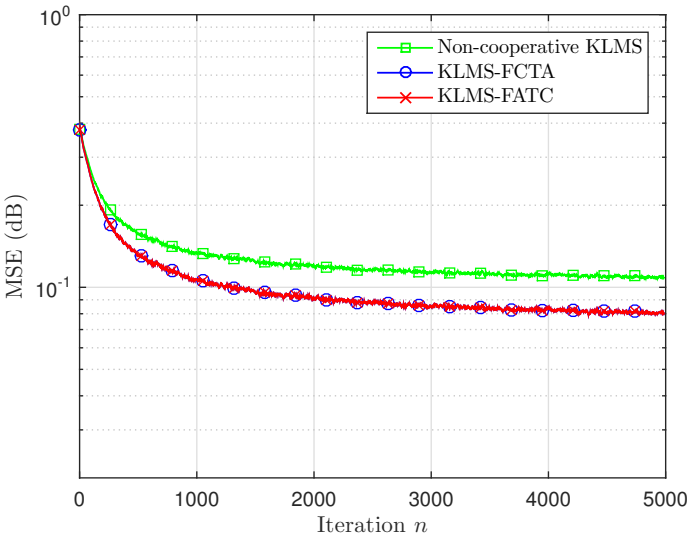
$$\begin{cases} v_{k,i} &= 1.1\exp(-|v_{k,i}|) + u_{k,i} \\ d_k(i) &= v_{k,i}^2 \end{cases} \tag{6.53}$$

where $u_{k,i}$ is the input signal. All input and output data pairs at 101×101 square grid were generated from the initial condition $v_{k,0} = 0.5$. The input $u_{k,i}$ was sampled from a zero-mean Gaussian distribution with standard deviation 0.25. The system output $d_k(i)$ was corrupted by an additive zero-mean white Gaussian noise with standard deviation $\sigma_{z,k} = 3$. The Laplacian kernel $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2/\xi)$ with kernel bandwidth $\xi = 0.35$ was considered. For all nodes, the step-sizes μ_k were set to 0.01, and the threshold of the CS criterion δ_0 was set to 0.3. A dictionary with 6 elements was then obtained [Richard 2009]. The coefficient $c_{\ell n}$ of matrix $\boldsymbol{C}$ was set to $|\mathcal{N}_n|^{-1}$ for all $n \in \mathcal{N}_\ell$. The combination matrix $\boldsymbol{A}$ simply averaged the estimates from the neighbors, namely, $a_{\ell n} = |\mathcal{N}_n|^{-1}$ for $\ell \in \mathcal{N}_n$.

The MSE learning curves in Figure 6.4(b) show that cooperative KLMS-FCTA and KLMS-FATC strategies perform better than the non-cooperative KLMS strategy. In addition, observe that the KLMS-FCTA and KLMS-FATC learning curves are almost identical.



(a) Network topology for experiment 1.



(b) MSE learning curves.

Figure 6.4: Experiment 1.

6.4.2 Example 2

As in Example 1, consider a network consisting of $N = 10$ nodes with the topology shown in Figure 6.5(a). The input signal at each node k and time instant i was a sequence of statistically independently vector [Vörös 2003] defined as:

$$\boldsymbol{u}_{k,i} = \begin{bmatrix} u_{k,i}(1) & u_{k,i}(2) \end{bmatrix}^\top \tag{6.54}$$

with correlated samples satisfying $u_{k,i}(1) = 0.5\, u_{k,i}(2) + v_{k,i}$. The second entry of $\boldsymbol{u}_{k,i}$ and $v_{k,i}$ were i.i.d. Gaussian samples with variance both equal to 0.035. We considered the linear system with memory defined by

$$\begin{aligned} y_{k,i} &= \boldsymbol{a}^\top \boldsymbol{u}_{k,i} - 0.2\, y_{k,i-1} + 0.35\, y_{k,i-2} \\ y_{k,0} &= 0, \;\; y_{k,-1} = 0 \end{aligned} \tag{6.55}$$

where $\boldsymbol{a} = [1 \; 0.5]^\top$ and the nonlinear Wiener function

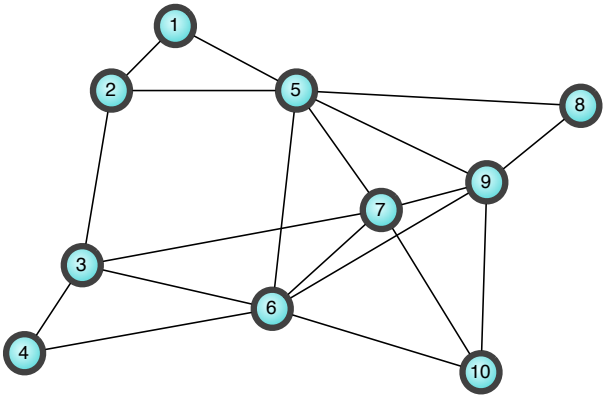
$$\varphi(y_{k,i}) = \begin{cases} \dfrac{y_{k,i}}{3[0.1 \; + \; 0.9\, y_{k,i}^2]^{1/2}} & \text{for } y_{k,i} \geq 0 \\ \dfrac{-y_{k,i}^2[1 - \exp(0.7 y_{k,i})]}{3} & \text{for } y_{k,i} < 0 \end{cases} \tag{6.56}$$

$$d_k(i) = \varphi(y_{k,i}) + z_{k,i}. \tag{6.57}$$

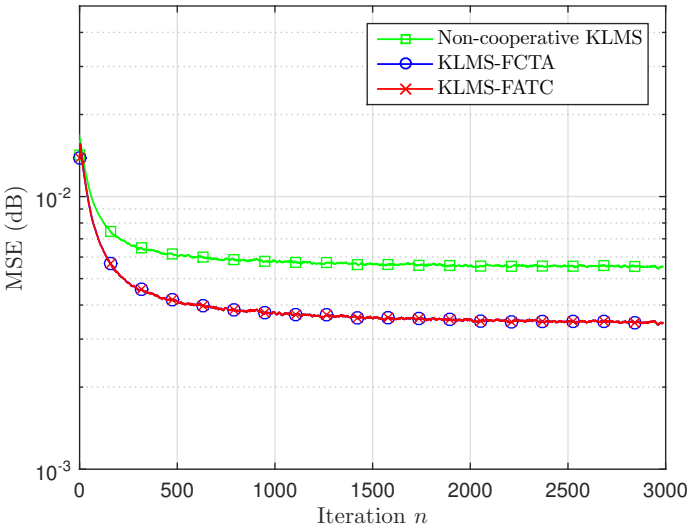
The output signal $d_k(i)$ was corrupted by a zero-mean i.i.d. Gaussian noise $z_{k,i}$ with variance $\sigma_{z_k}^2 = 0.09$. The initial condition $y_{k,1} = 0$ was considered. The bandwidth ξ of the Gaussian kernel $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2 / 2\xi^2)$ was set to 0.15, and the step-size μ_k were set to 0.05. The entries $c_{\ell n}$ of $\boldsymbol{C}$ were set to $|\mathcal{N}_n|^{-1}$ for all $n \in \mathcal{N}_\ell$. The threshold of the coherence criterion δ_0 was set to 0.3, which led to a 17-elements dictionary that was fixed during all the experiment. The combination matrix $\boldsymbol{A}$ simply averaged the estimates from the neighbors, namely, $a_{\ell n} = |\mathcal{N}_n|^{-1}$ for $\ell \in \mathcal{N}_n$. Again, the cooperative KLMS-FCTA and KLMS-FATC diffusion algorithms led to better performances than the non-cooperative KLMS algorithm.

6.5 Conclusion

Distributed learning over networks has become an active topic of research in the last decade. Adaptive networks are suitable for decentralized inference tasks, e.g., to monitor complex natural phenomena or infrastructure. Most of works have studied distributed estimation methods of linear regression models. However, there are many important applications that deal with nonlinear parametric models to be fitted, in a collaborative manner, over the area covered by a network. In this chapter, we derived functional counterparts of the adapt-then-combine and combine-then-adapt diffusion strategies in RKHS to address dynamic field inference problems. These diffusion strategies with KLMS algorithm allow to perform online learning of nonlinear fitting models. Their efficiency was illustrated with simulation results.



(a) Network topology for experiment 2.



(b) MSE learning curves.

Figure 6.5: Experiment 2.

Chapter 7

Conclusions and perspectives

Contents

7.1 Thesis summary

This thesis presented the use of KLMS algorithm in different contexts and, in each case, an analysis of its convergence behavior in the mean and mean-square-error sense. KLMS algorithms, with their simplicity and robustness, allow us to consider them as a benchmark of performance for KAF. Although various kernel adaptive filters with distinct features were continuously proposed in the literature, it is obviously insufficient to only develop KAF algorithms for possible practical applications without completely exploring their performance. Accordingly, we investigated the theoretical performance of real-valued monokernel LMS algorithm in nonstationary environment, real-valued multikernel LMS algorithm, complex multikernel LMS algorithm and diffusion adaption over networks with KLMS algorithm. This thesis thus provides a theoretical tool to analyze, design and compare the performance of KAF algorithms with different characteristics within the framework of KLMS algorithms.

In Chapter 2, we started by recalling some basic concepts of classic linear adaptive filtering, especially the milestones for convergence analysis of LMS algorithm. In order to understand the basic principles of kernel-based methods thoroughly, the concepts and properties of RKHS and kernel function were provided. Then, we introduced the necessary sparsification criteria for online dictionary learning with KAF. Finally, three existing kernel adaptive filtering algorithms were presented, including KAP algorithm, KNLMS algorithm and KRLS algorithm.

Most of the existing strategies for dictionary update are only able to incorporate new candidates into dictionaries. In other words, the obsolete and invalid kernel functions cannot be automatically eliminated from the dictionary in an online way, which may degrade the performance of KAF particularly in a nonstationary environment. In Chapter 3, we studied the convergence behavior of the Gaussian monokernel LMS in the case where the statistics of the elements of dictionary only partially match the statistics of the time-varying input data. The theoretical analysis emphasized the necessity of updating the dictionary in an online way, by simultaneously discarding the obsolete elements and including the appropriate ones. We proposed the KLMS algorithm with ℓ_1-norm regularization to achieve

the task of updating dictionary in real-time. The stability of the FOBOS-KLMS-Ω in the mean was also provided to guarantee its effectiveness in online processing.

Following the theoretical studies of monokernel algorithms dealing with the nonstationary input signals in Chapter 3, we firstly introduced two types of single-input and multiple-input multikernel LMS algorithms with more degrees of freedom, which generally leads to improved performance over monokernel methods. In fact the SI-MKLMS offers much more flexibility and still is a particular case of mono-kernel LMS algorithm. Then we focused on deriving the analytical model of stochastic convergence behavior of MI-MKLMS algorithm. Finally, the simulation results not only demonstrated the accuracy of our theoretical analysis, but also the respective characteristics of SI-MKLMS and MI-MKLMS algorithms.

As complex data are able to intrinsically preserve the phase information, the complex KAF was investigated in Chapter 5. After providing the preliminaries of complex kernel function and RKHS, two complex monokernel adaptive filtering algorithms were presented. Then, ACKLMS algorithm for non-circular input was proposed as a basic tool for complex multikernel adaptive filtering. In order to investigate the stochastic behavior of ACKLMS algorithm, we conducted its mean weight error analysis and mean-square error analysis. The numerical experiments validated our theoretical analyses.

Within the framework of distributed learning over sensor networks, Chapter 6 considered nonlinear diffusion adaptation in RKHS. By combining the basic monokernel LMS algorithm with two well known diffusion strategies, we developed FATC/FCTA algorithms and their parametric vector expressions. Moreover, we discussed the stability of proposed algorithms in the mean sense. The numerical results showed that the performances of cooperative FCTA and FATC diffusion algorithms outperformed the non-cooperative KLMS.

7.2 Perspectives

In this thesis, we introduced theoretical analyses of KLMS algorithms in different contexts, in an systematic and comprehensive way. However, there are still some interesting problems to be further studied. We will consider them as further research possibilities summarized below.

- Gaussian kernel function is extensively used with most of the kernel-based methods. The kernel bandwidth ξ directly determines the sharpness of Gaussian function fitting the objective function, which finally influences the performance of KAF algorithms. However, how to realize the adaptive adjustment of the Gaussian kernel bandwidth in an online manner is of particular interest and an open problem until now. That is to say that the online mode selection of KAF is a very challenging and interesting problem for practical implementations in case of lack of prior statistical information, and thus should not be bypassed in future works.

- Distributed signal processing on sensor networks is a promising research trend with the development of wireless sensor technologies and various of actual demands such as monitoring complex environment, nonlinear target tracking, etc. In chapter 6, we

made an attempt to combine the KLMS algorithm with diffusion adaptation strategies. We will address the problem of dictionary learning at each node to circumvent the drawbacks of KLMS. The convergence of FCTA and FATC algorithms in mean and MSE sense shall be analyzed in detail with this setting.

- The Cayley-Dickson hypercomplex number system and its algebraic properties were established as the standard extension of real, complex, quaternion, octonion and sedenion, etc. as mentioned in Chapter 1. Meanwhile, the m-dimensional ($m = 2^n, n \in \mathbb{Z}_+$) C-D hypercomplex valued KLMS algorithm was proposed based on the algebraic translation. However, there is no corresponding theoretical analysis of C-D hypercomplex valued KLMS algorithm. Although an analysis of KLMS in C-D hypercomplex domain intuitively seems to be extremely complicated, once it will be available, it will be viewed as an ultimate universal analysis ranging from the simplest real-valued to 2^m-dimensional hypercomplex valued.

- In light of the development of analysis of LMS algorithm, it is also a challenging possibility of analyzing the stochastic convergence behavior of the kernel least mean fourth (KLMF) algorithm in future work.

- Last but not least, we shall consider how all the derived analytical models are able to be applied for specific design of kernel adaptive filters dedicated to real implementations such as nonlinear acoustic echo cancellation, nonlinear channel identification system, etc.

Bibliography

[Aronszajn 1950] N. Aronszajn. *Theory of reproducing kernels.* Trans. Amer. Math. Soc., vol. 68, no. 3, pages 337–404, 1950. (Cited on pages 13, 15 and 53.)

[Bach 2004] F.R. Bach, G.R.G. Lanckriet and M.I. Jordan. *Multiple kernel learning, conic duality, and the SMO algorithm.* In Proceedings of the Twenty-First international conference on Machine learning, New York, USA, 2004. ACM. (Cited on pages 3 and 51.)

[Bauschke 2011] H.H. Bauschke and P.L. Combettes. Convex analysis and monotone operator theory in hilbert spaces. Springer, 2011. (Cited on pages 44 and 45.)

[Beck 2009] A. Beck and M. Teboulle. *A fast iterative shrinkage-thresholding algorithm for linear inverse problems.* SIAM Journal on Imaging Sciences, vol. 2, no. 1, pages 183–202, 2009. (Cited on page 44.)

[Bergamo 2012] Y.P. Bergamo and C.G. Lopes. *Scalar field estimation using adaptive networks.* In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3565–3568, 2012. (Cited on page 84.)

[Bershad 2008] N.J. Bershad, J.-C.M. Bermudez and J.-Y. Tourneret. *An affine combination of two LMS adaptive filters—transient mean-square analysis.* IEEE Transactions on Signal Processing, vol. 56, no. 5, pages 1853–1864, 2008. (Cited on page 1.)

[Bouboulis 2010] P. Bouboulis. *Wirtinger's calculus in general Hilbert spaces.* Rapport technique Available: http://arxiv.org/abs/1005.5170, University of Athens, 2010. (Cited on page 69.)

[Bouboulis 2011] P. Bouboulis and S. Theodoridis. *Extension of Wirtinger's calculus to reproducing kernel Hilbert spaces and the complex kernel LMS.* IEEE Transactions on Signal Processing, vol. 59, no. 3, pages 964–978, 2011. (Cited on pages 3, 69, 71 and 72.)

[Bouboulis 2012] P. Bouboulis, S. Theodoridis and M. Mavroforakis. *The augmented complex kernel LMS.* IEEE Transactions on Signal Processing, vol. 60, no. 9, pages 4962–4967, 2012. (Cited on pages 3, 69 and 74.)

[Camps-Valls 2007] G. Camps-Valls, J.L. Rojo-Alvarez and M. Martinez-Ramon, editeurs. Kernel methods in bioengineering, signal and image processing. IGI Global, 2007. (Cited on page 4.)

[Camps-Valls 2009] G. Camps-Valls and L. Bruzzone, editeurs. Kernel methods for remote sensing data analysis. Wiley, 2009. (Cited on page 4.)

[Candès 2008] E.J. Candès, M.B. Wakin and S.P. Boyd. *Enhancing sparsity by reweighted ℓ_1 minimization.* Journal of Fourier Analysis and Applications, vol. 14, no. 5, pages 877–905, 2008. (Cited on page 45.)

[Candès 2011] E.J. Candès, Y.C. Eldar, D. Needell and P. Randall. *Compressed sensing with coherent and redundant dictionaries.* Applied and Computational Harmonic Analysis, vol. 31, no. 1, pages 59–73, 2011. (Cited on page 30.)

[Cawley 2004] G.C. Cawley, N.C. Talbot, R.J. Foxall, S.R. Dorling and D.P. Mandic. *Heteroscedastic kernel ridge regression.* Neurocomputing, vol. 57, pages 105–124, 2004. (Cited on page 2.)

[Chen 2009] Y. Chen, Y. Gu and A.O. Hero. *Sparse LMS for system identification.* In IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 3125–3128, 2009. (Cited on pages 30 and 45.)

[Chen 2012a] B. Chen, S. Zhao, S. Seth and J.C. Príncipe. *Online efficient learning with quantized KLMS and L1 regularization.* In IEEE International Joint Conference on Neural Networks (IJCNN), pages 1–6, 2012. (Cited on pages 29, 30 and 34.)

[Chen 2012b] B. Chen, S. Zhao, P. Zhu and J.C. Príncipe. *Quantized kernel least-mean-square algorithm.* IEEE Transactions on Neural Networks and Learning Systems, vol. 23, no. 1, pages 22–32, 2012. (Cited on pages 3 and 31.)

[Chen 2013a] B. Chen, S. Zhao, P. Zhu and J.C. Príncipe. *Quantized kernel recursive least squares algorithm.* IEEE Transactions on Neural Networks and Learning Systems, vol. 24, no. 9, pages 1484–1491, 2013. (Cited on page 2.)

[Chen 2013b] J. Chen, C. Richard and P. Honeine. *Nonlinear unmixing of hyperspectral data based on a linear-mixture/nonlinear-fluctuation model.* IEEE Transactions on Signal Processing, vol. 61, no. 2, pages 480–492, 2013. (Cited on page 4.)

[Chen 2013c] J. Chen, C. Richard and P. Honeine. *Nonlinear unmixing of hyperspectral data based on a linear-mixture/nonlinear-fluctuation model.* IEEE Transactions on Signal Processing, vol. 61, no. 2, pages 480–492, 2013. (Cited on page 67.)

[Chen 2014a] J. Chen, W. Gao, C. Richard and J.-C.M. Bermudez. *Convergence analysis of kernel LMS algorithm with pre-tuned dictionary.* In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7243–7247, 2014. (Cited on pages 56, 57 and 92.)

[Chen 2014b] J. Chen, C. Richard and P. Honeine. *Nonlinear estimation of material abundances in hyperspectral images with ℓ_1-norm spatial regularization.* IEEE Transactions on Geoscience and Remote Sensing, vol. 52, no. 5, pages 2654–2665, 2014. (Cited on page 4.)

[Cucker 2001] F. Cucker and S. Smale. *On the mathematical foundations of learning.* Bulletin of the American Mathematical Society, vol. 31, no. 1, pages 1–49, 2001. (Cited on page 31.)

[Dobigeon 2014] N. Dobigeon, J.-Y. Tourneret, C. Richard, J.-C.M. Bermudez, S. McLaughlin and A.O. Hero. *Nonlinear unmixing of hyperspectral images: Models and algorithms*. IEEE Signal Processing Magazine, vol. 31, no. 1, pages 82–94, 2014. (Cited on page 4.)

[Duchi 2009] J. Duchi and Y. Singer. *Efficient online and batch learning using forward backward splitting*. Journal of Machine Learning Research, vol. 10, pages 2899–2934, 2009. (Cited on page 44.)

[Duda 2012] R.O. Duda, P.E. Hart and D.G. Stork. Pattern classification. John Wiley & Sons, 2012. (Cited on page 1.)

[Engel 2004] Y. Engel, S. Mannor and R. Meir. *The kernel recursive least squares*. IEEE Transactions on Signal Processing, vol. 52, no. 8, pages 2275–2285, 2004. (Cited on pages 2 and 23.)

[Gao 2013] W. Gao, J. Chen, C. Richard, J. Huang and R. Flamary. *Kernel LMS algorithm with forward-backward splitting for dictionary learning*. In IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 5735–5739. IEEE, 2013. (Cited on pages 30 and 34.)

[Gao 2014a] W. Gao, J. Chen, C. Richard and J. Huang. *Online dictionary learning for kernel LMS*. IEEE Transactions on Signal Processing, vol. 62, no. 11, pages 2765–2777, 2014. (Cited on page 70.)

[Gao 2014b] W. Gao, C. Richard, J.-C.M. Bermudez and J. Huang. *Convex combinations of kernel adaptive filters*. In IEEE International Workshop on Machine Learning for Signal Processing (MLSP), pages 1–5, 2014. (Cited on page 3.)

[Gonen 2011] M. Gonen and E. Alpaydin. *Multiple kernel learning algorithms*. The Journal of Machine Learning Research, vol. 12, pages 2211–2268, 2011. (Cited on pages 3 and 51.)

[Guestrin 2004] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin and S. Madden. *Distributed regression: an efficient framework for modeling sensor network data*. In IEEE International Symposium on Information Processing in Sensor Networks (IPSN), pages 1–10, USA, 2004. (Cited on page 84.)

[Haykin 1991] S. Haykin. Adaptive filter theory. Prentice-Hall, New Jersey, 2nd édition, 1991. (Cited on pages 2, 8, 12, 13, 30 and 33.)

[Haykin 1999] S. Haykin. Neural networks: A comprehensive foundation. Prentice Hall, Englewood Cliffs, NJ, 1999. (Cited on page 21.)

[Honeine 2007] P. Honeine, C. Richard and J.-C.M. Bermudez. *On-line nonlinear sparse approximation of functions*. In IEEE International Symposium on Information Theory (ISIT), pages 956–960, 2007. (Cited on pages 2 and 26.)

[Honeine 2009] P. Honeine, C. Richard, J.-C.M. Bermudez, H. Snoussi, M. Essoloh and F. Vincent. *Functional estimation in Hilbert space for distributed learning in wireless sensor networks.* In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 2861–2864, 2009. (Cited on page 84.)

[Ishida 2013] T. Ishida and T. Tanaka. *Multikernel adaptive filters with multiple dictionaries and regularization.* In IEEE Signal and Information Processing Association Annual Summit and Conference (APSIPA), Kaohsiung, Taiwan, 2013. (Cited on page 3.)

[Kar 2009] S. Kar and J.M.F. Moura. *Distributed consensus algorithms in sensor networks with imperfect communication: Link failures and channel noise.* IEEE Transactions on Signal Processing, vol. 57, no. 1, pages 355–369, 2009. (Cited on pages 83 and 84.)

[Kloft 2009] M. Kloft, U. Brefeld, S. Sonnenburg, P. Laskov, K.R. Müller and A. Zien. *Efficient and accurate ℓ_p-norm multiple kernel learning.* Advances in Neural Information Processing Systems, vol. 22, no. 22, pages 997–1005, 2009. (Cited on pages 3 and 51.)

[Kolmogorov 1957] A.N. Kolmogorov. *On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition.* Doklady Akademii Nauk USSR, vol. 114, pages 953–956, 1957. (Cited on page 21.)

[Kuh 2009] A. Kuh and D.P. Mandic. *Applications of complex augmented kernels to wind profile prediction.* In IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 3581–3584, 2009. (Cited on page 4.)

[Kung 2009] S.Y. Kung. *Kernel approaches to unsupervised and supervised machine learning.* In Advances in Multimedia Information Processing-PCM 2009, pages 1–32. Springer, 2009. (Cited on pages 3 and 69.)

[Lanckriet 2004] G.R.G Lanckriet, N. Cristianini, L.El Ghaoui, P. Bartlett and M. Jordan. *Learning the kernel matrix with semi-definite programming.* Journal of Machine Learning Research, vol. 5, pages 27–72, 2004. (Cited on page 52.)

[Liu 2008a] W. Liu, P.P. Pokharel and J.C. Príncipe. *The kernel least-mean-square algorithm.* IEEE Transactions on Signal Processing, vol. 56, no. 2, pages 543–554, 2008. (Cited on pages 3 and 31.)

[Liu 2008b] W. Liu and J.C. Príncipe. *Kernel affine projection algorithms.* EURASIP Journal on Advances in Signal Processing, vol. 2008, 2008. (Cited on page 2.)

[Liu 2009a] W. Liu, I.M. Park and J.C. Príncipe. *An information theoretic approach of designing sparse kernel adaptive filters.* IEEE Transactions on Neural Networks, vol. 20, no. 12, pages 1950–1961, 2009. (Cited on page 24.)

[Liu 2009b] W. Liu, I.M. Park, Y. Wang and J.C. Príncipe. *Extended kernel recursive least squares algorithm.* IEEE Transactions on Signal Processing, vol. 57, no. 10, pages 3801–3814, 2009. (Cited on page 2.)

[Liu 2010] W. Liu, J.C. Príncipe and S. Haykin. Kernel adaptive filtering: A comprehensive introduction. Wiley, New Jersey, 2010. (Cited on pages 2 and 23.)

[Lopes 2007a] C. G. Lopes and A. H. Sayed. *Diffusion least-mean squares over adaptive networks.* In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), volume 3, pages 917–920, 2007. (Cited on page 83.)

[Lopes 2007b] C.G. Lopes and A.H. Sayed. *Incremental adaptive strategies over distributed networks.* IEEE Transactions on Signal Processing, vol. 55, no. 8, pages 4064–4077, 2007. (Cited on page 83.)

[Luenberger 1979] D.G. Luenberger. Introduction dynamic systems: Theory, models & applications. Jonh Wiley & Sons, Inc., Stanford, CA, 1979. (Cited on page 63.)

[Mandic 2004] D.P. Mandic. *A generalized normalized gradient descent algorithm.* IEEE Signal Processing Letters, vol. 2, pages 115–118, 2004. (Cited on page 38.)

[Mandic 2009] D.P. Mandic, S. Javidi, S.L. Goh, A. Kuh and K. Aihara. *Complex-valued prediction of wind profile using augmented complex statistics.* Renewable Energy, vol. 34, no. 1, pages 196–201, 2009. (Cited on pages 3 and 69.)

[Martins 2011] A. Martins, N.A. Smith, E.P. Xing, P. Aguiar and M. Figueiredo. *Online learning of structured predictors with multiple kernels.* In Proc. of International Conference on Artificial Intelligence and Statistics, pages 507–515, 2011. (Cited on pages 3 and 51.)

[Mathews 2000] V.J. Mathews and G.L. Sicuranze. Polynomial signal processing. John Wiley & Sons, New York, NY, 2000. (Cited on page 21.)

[Mercer 1909] J. Mercer. *Functions of positive and negative type, and their connection with the theory of integral equations.* Philosophical Transactions of the Toyal Society of London (A), vol. 29, pages 415–446, 1909. (Cited on page 14.)

[Micchelli 1986] C.A. Micchelli. Algebraic aspects of interpolation. IBM Thomas J. Watson Research Division, 1986. (Cited on page 18.)

[Minkoff 2001] 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, vol. 49, no. 5, page 1109, 2001. (Cited on pages 35, 58 and 91.)

[Mizoguchi 2013] T. Mizoguchi and I. Yamada. *An algebraic real translation of hypercomplex linear systems and its application to adaptive filtering.* In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 6181–6185, 2013. (Cited on page 4.)

[Mizoguchi 2014] T. Mizoguchi and I. Yamada. *An algebraic translation of Cayley-Dickson linear systems and its applications to online learning.* IEEE Transactions on Signal Processing, vol. 62, no. 6, pages 1438–1453, 2014. (Cited on page 4.)

[Moore 1916] E.H. Moore. *On properly positive Hermitian matrices.* Bull. Amer. Math. Soc, vol. 23, no. 59, pages 66–67, 1916. (Cited on page 16.)

[Murakami 2010] Y. Murakami, M. Yamagishi, M. Yukawa and I. Yamada. *A sparse adaptive filtering using time-varying soft-thresholding techniques.* In IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 3734–3737, 2010. (Cited on page 30.)

[Nam 1994] S.W. Nam and E.J. Powers. *Application of higher order spectral analysis to cubically nonlinear system identification.* IEEE Signal Processing Magazine, vol. 42, no. 7, pages 2124–2135, 1994. (Cited on page 21.)

[Narendra 1990] K.S. Narendra and K. Parthasarathy. *Identification and control of dynamical systems using neural networks.* IEEE Transactions on Neural Networs, vol. 1, no. 1, pages 3–27, 1990. (Cited on page 38.)

[Nguyen-Tuong 2012] D. Nguyen-Tuong and J. Peters. *Online kernel-based learning for task-space tracking robot control.* IEEE Transactions on Neural Networks and Learning Systems, vol. 23, no. 9, pages 1417–1425, 2012. (Cited on page 4.)

[Nguyen 2005] X. Nguyen, M.J. Wainwright and M.I. Jordan. *Nonparametric decentralized detection using kernel methods.* IEEE Transactions on Signal Processing, vol. 53, no. 11, pages 4053–4066, 2005. (Cited on pages 83 and 84.)

[Nikias 1993] C.L. Nikias and A.P. Petropulu. Higher-order spectra analysis - a nonlinear signal processing pramework. Prentice Hall, Englewood Cliffs, NJ, 1993. (Cited on page 21.)

[Ogunfunmi 2015] T. Ogunfunmi and T. Paul. *The quarternion maximum correntropy clgorithm.* IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 62, no. 6, pages 598–602, 2015. (Cited on page 4.)

[Omura 1965] J. Omura and T. Kailath. *Some useful probability distributions.* Rapport technique 7050-6, Stanford Electronics Laboratories, Stanford University, Stanford, California, USA, 1965. (Cited on pages 33 and 58.)

[Parreira 2012] W.D. Parreira, J.-C.M. Bermudez, C. Richard and J.-Y. Tourneret. *Stochastic behavior analysis of the Gaussian kernel-least-mean-square algorithm.* IEEE Transactions on Signal Processing, vol. 60, no. 5, pages 2208–2222, 2012. (Cited on pages 29, 30, 33, 34, 35, 36, 38, 47, 56, 58, 61, 69, 77 and 91.)

[Paul 2015] T.K. Paul and T. Ogunfunmi. *A kernel adaptive algorithm for quaternion-valued inputs.* IEEE Transactions on Neural Networks and Learning Systems, page to appear, 2015. (Cited on page 4.)

[Platt 1991] J. Platt. *A resource-allocating network for function interpolation.* Neural Computation, vol. 3, no. 2, pages 213–225, 1991. (Cited on page 23.)

[Pokharel 2013] R. Pokharel, S. Seth and J.C. Príncipe. *Mixture kernel least mean square.* In IEEE International Joint Conference on Neural Networks (IJCNN), pages 1–7, Dallas, USA, 2013. (Cited on pages 3 and 70.)

[Predd 2005] J.B. Predd, S.R. Kulkarni and H.V. Poor. *Regression in sensor networks: Training distributively with alternating projections.* In Proc. SPIE Conf. Advanced Signal Processing Algorithm, Architectures, and Implementations XV, San Diego, CA, 2005. (Cited on page 84.)

[Predd 2006] J.B. Predd, S.B. Kulkarni and H.V. Poor. *Distributed learning in wireless sensor networks.* IEEE Signal Processing Magazine, vol. 23, no. 4, pages 56–69, 2006. (Cited on page 84.)

[Rakotomamonjy 2008] A. Rakotomamonjy, F. Bach, S. Canu and Y. Granvalet. *SimpleMKL.* Journal of Machine Learning Research, vol. 9, pages 2491–2521, 2008. (Cited on pages 3, 51 and 67.)

[Richard 2005] C. Richard. *Filtrage adaptatif non-linéaire par méthodes de gradient stochastique court-terme à noyau.* In Actes du 20e Colloque GRETSI sur le Traitement du Signal et des Images, 2005. (Cited on page 3.)

[Richard 2009] C. Richard, J.-C.M. Bermudez and P. Honeine. *Online prediction of time series data with kernels.* IEEE Transactions on Signal Processing, vol. 57, no. 3, pages 1058–1067, 2009. (Cited on pages 2, 24, 31, 32, 85, 86 and 93.)

[Richard 2012] C. Richard and J.-C M. Bermudez. *Closed-form conditions for convergence of the Gaussian kernel-least-mean-square algorithm.* In IEEE Asilomar Conference on Signals, Systems, and Computers, pages 1797–1801, 2012. (Cited on page 29.)

[Riesz 1907] F. Riesz. *Sur une espèce de géométrie analytique des systèmes de fonctions sommables.* CR Acad. Sci. Paris, vol. 144, pages 1409–1411, 1907. (Cited on page 15.)

[Rosipal 2002] R. Rosipal and L.J. Trejo. *Kernel partial least squares regression in reproducing kernel Hilbert space.* The Journal of Machine Learning Research, vol. 2, pages 97–123, 2002. (Cited on page 2.)

[Russell 1995] S. Russell and P. Norvig. *Artificial intelligence: a modern approach.* 1995. (Cited on page 1.)

[Sayed 2003] A.H. Sayed. Fundamentals of adaptive filtering. Wiley, New York, 2003. (Cited on pages 2, 33, 35, 58 and 61.)

[Sayed 2008] A.H. Sayed. Adaptive filters. John Wiley & Sons, 2008. (Cited on page 91.)

[Sayed 2013a] A.H. Sayed. *Diffusion adaptation over networks*. E-Reference Signal Processing, R. Chellapa and S. Theodoridis, editors, Elsevier, 2013. (Cited on page 84.)

[Sayed 2013b] A.H. Sayed, S.-Y Tu, J. Chen, X. Zhao and Z.J. Towfic. *Diffusion strategies for adaptation and learning over networks: An examination of distributed strategies and network behavior*. IEEE Signal Processing Magazine, vol. 30, no. 3, pages 155–171, 2013. (Cited on page 84.)

[Schetzen 1980] M. Schetzen. The Volterra and Wiener theory of the nonlinear systems. Wiley, New York, NY, 1980. (Cited on page 21.)

[Schölkopf 1997] B. Schölkopf, A. Smola and K.-R. Müller. *Kernel principal component analysis*. In Artificial Neural Networks—ICANN'97, pages 583–588. Springer, 1997. (Cited on page 2.)

[Schölkopf 2000] B. Schölkopf, R. Herbrich and R. Williamson. *A generalized representer theorem*. Rapport technique NC2-TR-2000-81, NeuroCOLT, Royal Holloway College, University of London, UK, 2000. (Cited on pages 13, 16 and 22.)

[Scholkopf 2001] B. Scholkopf and A.J. Smola. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press, 2001. (Cited on pages 2 and 16.)

[Shawe-Taylor 2004] J. Shawe-Taylor and N. Cristianini. Kernel methods for pattern analysis. Cambridge university press, 2004. (Cited on pages 13 and 15.)

[Slavakis 2008] K. Slavakis and S. Theodoridis. *Sliding window generalized kernel affine projection algorithm using projection mappings*. EURASIP Journal on Advances in Signal Processing, vol. 2008, 2008. (Cited on pages 2 and 3.)

[Slavakis 2010] K. Slavakis, Y. Kopsinis and S. Theodoridis. *Adaptive algorithm for sparse system identification using projections onto weighted ℓ_1 balls*. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 3742–3745, 2010. (Cited on page 30.)

[Slavakis 2012] K. Slavakis, P. Bouboulis and S. Theodoridis. *Adaptive multiregression in reproducing kernel Hilbert spaces: The multiaccess MIMO channel case*. IEEE Transactions on Neural Networks and Learning Systems, vol. 23, no. 2, pages 260–276, 2012. (Cited on page 3.)

[Slavakis 2013] K. Slavakis, P. Bouboulis and S. Theodoridis. *Online learning in reproducing kernel Hilbert spaces*. Signal Processing Theory and Machine Learning, pages 883–987, 2013. (Cited on pages 3, 13 and 24.)

[Sonnenburg 2006] S. Sonnenburg, G. Rätsch, C. Schäfer and B. Schölkopf. *Large scale multiple kernel learning*. Journal of Machine Learning Research, vol. 7, pages 1531–1565, 2006. (Cited on pages 3 and 51.)

[Srivastava 2011] K. Srivastava and A. Nedic. *Distributed asynchronous constrained stochastic optimization.* IEEE J. Sel. Topics Signal Process., vol. 5, no. 4, pages 772–790, 2011. (Cited on page 84.)

[Steinwart 2006] I. Steinwart, D. Hush and C. Scovel. *An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels.* IEEE Transactions on Information Theory, vol. 52, no. 10, pages 4635–4643, 2006. (Cited on page 72.)

[Steven 2006] V.V. Steven, J. Vía and I. Santamaría. *A sliding-window kernel RLS algorithm and its application to nonlinear channel identification.* In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 789–792, 2006. (Cited on page 2.)

[Steven 2012] V.V. Steven, M.l Lázaro-Gredilla and I. Santamaría. *Kernel recursive least-squares tracker for time-varying regression.* IEEE Transactions on Neural Networks and Learning Systems, vol. 23, no. 8, pages 1313–1326, 2012. (Cited on page 2.)

[Sutton 1998] R.S. Sutton and A.G. Barto. Reinforcement learning: An introduction. MIT press Cambridge, 1998. (Cited on page 1.)

[Takizawa 2015] M.-A. Takizawa and M. Yukawa. *Adaptive nonlinear estimation based on parallel projection along affine subspaces in reproducing kernel Hilbert space.* IEEE Transactions on Signal Processing, vol. 63, no. 16, pages 4257–4269, 2015. (Cited on page 3.)

[Theodoridis 2011] S. Theodoridis, K. Slavakis and I. Yamada. *Adaptive learning in a world of projections.* IEEE Signal Processing Magazine, vol. 28, no. 1, pages 97–123, 2011. (Cited on page 3.)

[Thomas 2013] P. Thomas and T. Ogunfunmi. *Study of the convergence behavior of the complex kernel least mean square algorithm.* IEEE Transactions on Neural Networks and Learning Systems, vol. 24, no. 9, pages 1349–1363, 2013. (Cited on page 69.)

[Tobar 2012a] F.A. Tobar, A. Kuh and D.P. Mandic. *A novel augmented complex valued kernel LMS.* In IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM), pages 473–476, 2012. (Cited on pages 3, 69 and 74.)

[Tobar 2012b] F.A. Tobar, A. Kuh and D.P. Mandic. *A novel augmented complex valued kernel LMS.* In IEEE 7th Sensor Array and Multichannel Signal Processing Workshop (SAM), pages 481–484, 2012. (Cited on page 4.)

[Tobar 2012c] F.A. Tobar and D.P. Mandic. *Multikernel least squares estimation.* In Proceedings of Sensor Signal Processing for Defence, London, UK, 2012. (Cited on pages 3, 52, 54 and 56.)

[Tobar 2013] F.A. Tobar and D.P. Mandic. *The quaternion kernel least squares.* In IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 6128–6132, 2013. (Cited on page 4.)

[Tobar 2014a] F.A. Tobar, S.-Y. Kung and D.P. Mandic. *Multikernel least mean square algorithm*. IEEE Transactions on Neural Networks and Learning Systems, vol. 25, no. 2, pages 265–277, 2014. (Cited on pages 3, 24, 52, 54 and 70.)

[Tobar 2014b] F.A. Tobar and D.P. Mandic. *A particle filtering based kernel HMM predictor*. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 7969–7973, 2014. (Cited on page 3.)

[Tobar 2014c] F.A. Tobar and D.P. Mandic. *Quaternion reproducing kernel Hilbert spaces: existence and uniqueness conditions*. IEEE Transactions on Information Theory, vol. 60, no. 9, pages 5736–5749, 2014. (Cited on page 4.)

[Tobar 2015] F. Tobar, P. Djuric and D.P. Mandic. *Unsupervised state-space modelling using reproducing kernels*. IEEE Transactions on Signal Processing, page to appear, 2015. (Cited on page 3.)

[Vörös 2003] J. Vörös. *Modeling and identification of Wiener systems with two-segment nonlinearities*. IEEE Transactions on Control Systems Technology, vol. 11, no. 2, pages 253–257, 2003. (Cited on pages 38, 64 and 94.)

[Widrow 1985] B. Widrow and S.D. Stearns. *Adaptive signal processing*. Englewood Cliffs, NJ, Prentice-Hall, Inc., vol. 1, 1985. (Cited on page 8.)

[Wiener 1958] N. Wiener. Nonlinear problems in random theory. Wiley, New York, NY, 1958. (Cited on page 21.)

[Xiao 2004] L. Xiao and S. Boyd. *Fast linear iterations for distributed averaging*. Systems & Control Letters, vol. 53, no. 1, pages 65–78, 2004. (Cited on page 84.)

[Xu 2007] X. Xu, D. Hu and X. Lu. *Kernel-based least squares policy iteration for reinforcement learning*. IEEE Transactions on Neural Networks and Learning Systems, vol. 18, no. 4, pages 973–992, 2007. (Cited on page 4.)

[Yukawa 2012] M. Yukawa. *Multikernel adaptive filtering*. IEEE Transactions on Signal Processing, vol. 60, no. 9, pages 4672–4682, 2012. (Cited on pages 3, 30, 34, 45, 52, 54 and 70.)

[Yukawa 2013] M. Yukawa and R. Ishii. *Online model selection and learning by multikernel adaptive filtering*. In IEEE European Signal Processing Conference (EUSIPCO), pages 1–5, Marrakech, Morocco, 2013. (Cited on pages 3, 52 and 54.)

[Zhu 2014] P. Zhu, B. Chen and J.C. Principe. *Learning nonlinear generative models of time series with a Kalman filter in RKHS*. IEEE Transactions on Signal Processing, vol. 62, no. 1, pages 141–155, 2014. (Cited on page 3.)

[Zou 2006] H. Zou. *The adaptive lasso and its oracle properties*. Journal of the American Statistical Association, vol. 101, no. 476, pages 1418–1429, 2006. (Cited on page 45.)