Set-Membership Adaptive Kernel NLMS Algorithms: Design and Analysis

André Flores, Rodrigo C. de Lamare

PII: S0165-1684(18)30238-X
DOI: https://doi.org/10.1016/j.sigpro.2018.07.007
Reference: SIGPRO 6871

To appear in: *Signal Processing*

Received date: 14 February 2018
Revised date: 21 May 2018
Accepted date: 10 July 2018

Please cite this article as: André Flores, Rodrigo C. de Lamare, Set-Membership Adaptive Kernel NLMS Algorithms: Design and Analysis, *Signal Processing* (2018), doi: https://doi.org/10.1016/j.sigpro.2018.07.007

Set-Membership Adaptive Kernel NLMS Algorithms: Design and Analysis

André Flores and Rodrigo C. de Lamare

Abstract

In the last decade, a considerable research effort has been devoted to developing adaptive algorithms based on kernel functions. One of the main features of these algorithms is that they form a family of universal approximation techniques, solving problems with nonlinearities elegantly. In this paper, we present data-selective adaptive kernel normalized least-mean square (KNLMS) algorithms that can increase their learning rate and reduce their computational complexity. In fact, these methods deal with kernel expansions, creating a growing structure also known as the dictionary, whose size depends on the number of observations and their innovation. The algorithms described herein use an adaptive step-size to accelerate the learning and can offer an excellent tradeoff between convergence speed and steady state, which allows them to solve nonlinear filtering and estimation problems with a large number of parameters without requiring a large computational cost. The data-selective update scheme also limits the number of operations performed and the size of the dictionary created by the kernel expansion, saving computational resources and dealing with one of the major problems of kernel adaptive algorithms. A statistical analysis is carried out along with a computational complexity analysis of the proposed algorithms. Simulations show that the proposed KNLMS algorithms outperform existing algorithms in examples of nonlinear system identification and prediction of a time series originating from a nonlinear difference equation.

Keywords

Adaptive algorithms, set-membership algorithms, data-selective techniques, kernel methods, statistical analysis.

I. Introduction

Adaptive filtering algorithms have been the focus of a great deal of research in the past decades and the machine learning community has embraced and further advanced the study of these methods. In fact, adaptive algorithms are often considered with linear structures, which limits their performance and does not draw attention

André Flores is with the Centre for Telecommunications Studies (CETUC), PUC-Rio, Rio de Janeiro, Brazil and Rodrigo C. de Lamare is with both CETUC and with the Department of Electronic Engineering, University of York, UK. Part of this work has been presented at the IEEE International Conference on Acoustics, Speech and Signal Processing 2017. The emails of the authors are andre.flores@cetuc.puc-rio.br and delamare@cetuc.puc-rio.br

to nonlinear problems that can be solved in various applications. In order to deal with nonlinear problems a family of nonlinear adaptive algorithms based on kernels has been developed. In particular, a kernel is a function that compares the similarity between two inputs and can be used for filtering, estimation and classification tasks. Kernel adaptive filtering (KAF) algorithms have been tested in many different scenarios and applications [1], [2], [3], [4], [5], showing very good results. One of the main advantages of KAF algorithms is that they are universal approximators [1], which gives them the ability to address complex and nonlinear problems. However, their computational complexity is much higher than their linear counterparts [1].

One of the first KAF algorithms to appear, which is widely adopted in the KAF family because of its simplicity, is the kernel least-mean square (KLMS) algorithm proposed in [6] and later extended in [7]. The KLMS algorithm has been inspired by the least-mean square (LMS) algorithm and, thanks to its good performance, led many researchers to work in the development of kernel versions of conventional adaptive algorithms. For instance, a kernel version of the NLMS algorithm has been proposed in [5] using a nonlinear regression approach for time series prediction. In [8], [9], the affine projection algorithm (APA) has been used as the basis of the derivation of kernel affine projection (KAP) algorithms. Adaptive projection algorithms using kernel techniques have been reported in [10], [11]. The recursive least squares algorithm (RLS) has been extended in [12], where the kernel recursive least squares (KRLS) has been described. Later, the authors of [13] proposed an extension of the KRLS algorithm and the use of multiple kernels has been studied in [14] and [15].

Previously reported kernel algorithms have to deal with kernel expansions, which increases significantly the computational cost. In other words, they create a growing structure, also called dictionary, where every new data input that arrives is employed to compute the estimate of the desired output. The natural problem that arises is that the time and computational cost required to compute a certain output could exceed the tolerable limits for an application. Several criteria to manage the growing structure of kernel algorithms have been proposed to solve this problem such as algorithms with fixed dictionary size as studied in [16], [17] and [18]. One of the most simple criteria is the novelty criterion (NC), presented in [19]. Specifically, NC establishes two thresholds to limit the size of the dictionary. Another method, the approximate linear dependency (ALD) has been proposed in [12] and verifies if a new input can be expressed as a linear combination of the elements stored before adding this input to the dictionary. The coherence criterion (CC) has been described in [5] also to limit the size of the dictionary based on the similarity of the inputs. A measure called surprise criterion (SC) has been presented in [20] to remove redundant data.

In this work, we present set-membership normalized kernel least-mean square (SM-KNLMS) adaptive algorithms, which have been initially reported in [21] and can provide a faster learning than existing kernel-based algorithms and limit the size of the dictionary without compromising performance. Unlike the equivalent set-theoretic approach in [11] the set-membership algorithms presented here exploit variable step sizes, which can lead to a faster

learning performance. Similarly to existing set-membership algorithms [22], [23], [24], [25], [26], [27], [28], the proposed SM-KNLMS algorithms are equipped with variable step sizes and perform sparse updates. We consider both centroid-based SM-KNLMS (C-SM-KNLMS) and nonlinear regression-based SM-KNLMS (NLR-KNLMS) algorithms, where the latter lends itself to statistical analysis [5]. Unlike existing kernel-based adaptive algorithms the proposed SM-KNLMS algorithms deal, in a natural way, with the kernel expansion because of the data selectivity based on error bounds that they implement. A statistical analysis of the NLR-SM-KNLMS algorithm along with the derivation of analytical formulas to predict the mean-square error (MSE), and an analysis of their computational cost are carried out. Simulations comparing the performance of the SM-KNLMS and existing algorithms for several scenarios are then conducted.

In summary, the contributions of this work are:

- The development of the proposed C-SM-KNLMS and NLR-SM-KNLMS algorithms.
- A statistical analysis of the NLR-SM-KNLMS algorithm and the development of analytical formulas to predict its performance.
- A simulation study of the proposed C-SM-KNLMS, NLR-SM-KNLMS and existing algorithms for several scenarios of interest.

This paper is organized as follows. In Section II, the principles of kernel methods and set-membership techniques are introduced. In Section III, we review set-membership adaptive algorithms and present the derivation of the proposed C-SM-KNLMS algorithm. Section IV presents the proposed NLR-SM-KNLMS algorithm. Section V details the statistical analysis of the NLR-SM-KNLMS algorithm and a comparison of the computational complexity of the proposed and existing algorithms. Section VI describes and discusses the simulation results and Section VII contains the conclusions of this work.

II. Principles of Kernel Methods and Set-Membership Techniques

Conventional adaptive algorithms work with linear structures, limiting the performance that they can achieve and constraining the number of problems that can be solved. Under this scope, a new family of nonlinear adaptive algorithms based on kernels has been developed [1]. The main objective of these algorithms is to learn an arbitrary input-output mapping based on a sequence of samples and a kernel. Basically, a kernel $\kappa(\cdot,\cdot)$ is a function that measures the similarity between two inputs and generally returns a real number. Several kernel functions are described in the literature [1]. Choosing a kernel function is important because it is equivalent to implicitly defining a feature space where the algorithms are performed. Let us now introduce two commonly used kernel functions. The first one is the Gaussian kernel, defined by

$$\kappa\left(\mathbf{x},\mathbf{x}'\right)=\exp\left(-\frac{\|\,\mathbf{x}-\mathbf{x}'\,\|^2}{2\nu^2}\right), \tag{1}$$

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

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

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

The relevant point about implementing kernel functions is that the scalar product can be implicitly computed in the feature space by a kernel evaluation, without explicitly using or even knowing the mapping applied to the data [29]. This means that there is no need to perform any operation on the high dimensional space, as long as the quantities are expressed as an inner product. This approach is known as the "kernel trick" and allows us to compute scalar products in spaces, where the computations are hard to perform. As a result, we avoid a significant increase of the computational complexity, which is one of the major problems that arises when working with high dimensional spaces. In particular we have

$$\kappa\left(\mathbf{x}, \mathbf{x}'\right) = \left\langle \kappa\left(\cdot, \mathbf{x}\right), \kappa\left(\cdot, \mathbf{x}'\right)\right\rangle. \tag{3}$$

To summarize, kernel adaptive algorithms map the data to a high-dimensional space through kernels. Then, linear methods can be applied on the transformed data to solve nonlinear problems.

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

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

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

We can extend linear models to nonlinear models by mapping the input data into a high-dimensional space. In order to perform this mapping, let us define a nonlinear transformation denoted by $\varphi : \mathbb{R}^{N} \rightarrow \mathbb{F}$, which maps the data in $\mathbb{R}^{N}$ to a high-dimensional feature space $\mathbb{F}$ that performs the nonlinear transformation. Applying the transformation stated before, we map both the input and the weights into the feature space which results in

$$\boldsymbol{\varphi}\left(i\right) = \varphi\left(\mathbf{x}\left(i\right)\right), \tag{5}$$

$$\boldsymbol{\omega}\left(i\right) = \varphi(\mathbf{w}\left(i\right)). \tag{6}$$

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

$$e\left(i\right) = d\left(i\right) - \boldsymbol{\omega}^{\mathrm{T}}\left(i\right)\boldsymbol{\varphi}\left(i\right). \tag{7}$$

The main idea behind set-membership algorithms is to model a function $\boldsymbol{\omega}\left(i\right)$, such that the magnitude of the estimated error defined by (7) is upper bounded by a quantity γ. Assuming that the value of γ is appropriately

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

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

$$\boldsymbol{\theta} = \bigcap_{\{\boldsymbol{\varphi},d\}\in\bar{\boldsymbol{S}}} \left\{\boldsymbol{\omega} \in \mathbb{F} \,/\, |d - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}| \leq \gamma\right\}. \tag{8}$$

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

$$\mathcal{H}(i) = \left\{\boldsymbol{\omega} \in \mathbb{F} \,/\, |d(i) - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\varphi}(i)| \leq \gamma\right\}. \tag{9}$$

It follows from (9) that, for each data pair, there exists an associated constraint set. The set containing the intersection of the constraint sets over all available time instants is called exact membership set and is given by the following equation:

$$\psi(i) = \bigcap_{k=0}^{i} \mathcal{H}(i). \tag{10}$$

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

III. Proposed Centroid-Based Set-Membership Kernel Normalized Least-Mean-Square Algorithm

In this section, we detail the derivation of the proposed C-SM-KNLMS algorithm, which is motivated by the possibility of of saving resources by not storing the zero coefficients in the parameter vector. In order to derive the C-SM-KLNMS algorithm, we check first if the previous solution is outside the constraint set, i.e.,

$$|d(i) - \boldsymbol{\omega}^{\mathrm{T}}(i)\,\boldsymbol{\varphi}(i)| > \gamma.$$

If the error exceeds the bound established, the algorithm performs an update so that the *a posteriori* estimated error lies in $\mathcal{H}(i)$.

The derivation of the C-SM-KNLMS algorithm corresponds to solving the following optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{\omega}(i+1)} ||\boldsymbol{\omega}(i+1) - \boldsymbol{\omega}(i)||^2 \\ &\text{subject to } \boldsymbol{\omega}(i+1) \in \mathcal{H}(i), \end{aligned} \tag{11}$$

where the *a posteriori* error $\xi_p(i)$ used to build the constraint set $\mathcal{H}(i)$ is given by

$$\xi_p(i) = d(i) - \boldsymbol{\omega}^{\mathrm{T}}(i+1)\boldsymbol{\varphi}(i) = \pm\gamma. \tag{12}$$

As mentioned in [1], the KNLMS update equation is given by

$$\boldsymbol{\omega}(i+1) = \boldsymbol{\omega}(i) + \frac{\mu(i)}{\varepsilon + ||\boldsymbol{\varphi}(i)||^2} e(i)\boldsymbol{\varphi}(i), \tag{13}$$

where $\mu(i)$ is the step-size that should be chosen to satisfy the constraints and ε is a small constant used to avoid numerical problems. Substituting (13) in (12) we arrive at:

$$\xi_p(i) = d(i) - \boldsymbol{\omega}^{\mathrm{T}}(i)\boldsymbol{\varphi}(i) - \frac{\mu(i)}{\varepsilon + ||\boldsymbol{\varphi}(i)||^2} e(i)\boldsymbol{\varphi}^{\mathrm{T}}(i)\boldsymbol{\varphi}(i) \tag{14}$$

Using (7) and replacing the dot products by kernel evaluations, the previous equation turns into:

$$\xi_p(i) = e(i) - \mu(i)e(i)\frac{\kappa(\mathbf{x}(i), \mathbf{x}(i))}{\varepsilon + \kappa(\mathbf{x}(i), \mathbf{x}(i))} = \pm\gamma. \tag{15}$$

Assuming that the constant ε is sufficiently small to ensure that

$$\frac{\kappa(\mathbf{x}(i), \mathbf{x}(i))}{\varepsilon + \kappa(\mathbf{x}(i), \mathbf{x}(i))} \approx 1, \tag{16}$$

then from Equation (15), we have

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

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

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

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

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

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

We can then compute $\boldsymbol{\omega}$ recursively as follows:

$$\begin{aligned} \boldsymbol{\omega}(i+1) =& \boldsymbol{\omega}(i-1) + \frac{\mu(i-1)e(i-1)}{\varepsilon + ||\boldsymbol{\varphi}(i-1)||^2}\boldsymbol{\varphi}(i-1) \\ &+ \frac{\mu(i)e(i)}{\varepsilon + ||\boldsymbol{\varphi}(i)||^2}\boldsymbol{\varphi}(i) \\ &\vdots \\ \boldsymbol{\omega}(i+1) =& \boldsymbol{\omega}(0) + \sum_{k=1}^{i} \frac{\mu(k)}{\varepsilon + ||\boldsymbol{\varphi}(k)||^2} e(k)\boldsymbol{\varphi}(k) \end{aligned} \tag{21}$$

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

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

The output $f(\boldsymbol{\varphi}(i+1)) = \boldsymbol{\omega}^{\mathrm{T}}(i+1)\boldsymbol{\varphi}(i+1)$ of the filter to a new input $\boldsymbol{\varphi}(i+1)$ can be computed as:

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

Using the kernel trick we obtain

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

where $\mu(k)$ is given by (20). Let us define a coefficient vector $\boldsymbol{a}(i)$ to store in each of its elements the following product:

$$[\boldsymbol{a}(i)]_k = \mu(k) e(k), \tag{25}$$

so that (24) becomes:

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

Eqs. (7), (20),(25), and (26) summarize the proposed C-SM-KNLMS algorithm. We set the initial values of $\boldsymbol{a}$ to zero. As new inputs arrive we can calculate the output of the system with (26). Then the error may be computed with (7) and if it exceeds the bound we compute the step-size with (20). The vector $\boldsymbol{a}(i)$ are updated with (25). Note that some coefficients may be zero due to the data selectivity of C-SM-KNLMS. We do not need to store the zero coefficients as they do not contribute to the output, resulting in saving of resources. This means that the dictionary at time instant i, denoted by $\boldsymbol{C}(i)$, has only m elements, with $m < i$. Each column of the dictionary, denoted by $\boldsymbol{c}_j$, contains the input that is used in the kth update. We can now rewrite (26) as follows:

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

This is an important result because it controls the growing network created by the algorithm [30]. In stationary environments the algorithm will limit the growing structure. Algorithm 1 summarizes the proposed C-SM-KNLMS algorithm. In particular, the computational complexity of C-SM-KNLMS grows over time with the increase of m, as illustrated by step 7 in Algorithm 1. However, we also note that the standard KNLMS algorithm exhibits such behavior with regards to the computational complexity. Unlike the standard KNLMS the proposed C-SM-KNLMS algorithm only performs an update when there is innovation according to the error bound, which limits the increment of m and consequently the increase in computational complexity.

Algorithm 1 Proposed C-SM-KNLMS algorithm

Initialization

1.Choose γ , ε and κ.

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

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

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

5.$m = 1$

Computation

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

%Compute the output

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

%Compute the error

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

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

%Compute the step-size

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

%Update the coefficients

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

%Store the new center

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

13. $m = m + 1$

14. **else**

15. $\mu(i) = 0$

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

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

18. **end if**

19.**end while**

IV. Proposed Nonlinear Regression-Based SM-KNLMS Algorithm

In this section, we follow a nonlinear regression approach as described in [5], [31], to develop an alternative SM-KNLMS algorithm, denoted NLR-SM-KNLMS algorithm.

Let us define a function $\psi(\cdot)$ on a feature space which, given an input vector $\mathbf{x}(i)$ generates the model output $\psi(\mathbf{x}(i))$. Our problem is now reduced to finding the function $\psi(\cdot)$ that minimizes the sum of the square error

between the desired response and the model output as described by

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

The representer theorem [32] states that the function $\psi\left(\cdot\right)$ can be expressed as a kernel expansion which depends on the available data, so that:

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

In order to derive the NLR-SM-KNLMS algorithm we need to solve the following optimization problem:

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

where $\boldsymbol{a} \in \mathbb{R}^m$ is the parameter vector to be computed, $\boldsymbol{d} \in \mathbb{R}^m$ is the vector with the desired signal and $\boldsymbol{K} \in \mathbb{R}^{m\times m}$ is the Gram matrix containing at each row i and each column j the kernel evaluations of the input data denoted by κ_{ij}, where we have

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

Let us now consider the case where we have a dictionary of size m so that $\boldsymbol{K} \in \mathbb{R}^{m\times m}$. Consider also a vector $\boldsymbol{\kappa_\delta}\left(i\right)$ that contains the kernel evaluations between the input data at time i and every input stored in the dictionary at time $i > m$ with $\boldsymbol{c}_j \neq \mathbf{x}\left(i\right)$ for $j = 1, \cdots, m$, given by

$$\boldsymbol{\kappa_\delta}\left(i\right) = \begin{bmatrix} \kappa\left(\mathbf{x}\left(i\right), \boldsymbol{c}_1\right) \\ \kappa\left(\mathbf{x}\left(i\right), \boldsymbol{c}_2\right) \\ \vdots \\ \kappa\left(\mathbf{x}\left(i\right), \boldsymbol{c}_{m+1}\right) \end{bmatrix}, \tag{32}$$

where $\boldsymbol{\kappa_\delta}\left(i\right)$ is used in the computation of an inner product with $\boldsymbol{a}(i+1) \in \mathbb{R}^{m+1}$. Using the minimum norm approach to obtain the NLR-SM-KNLMS algorithm, the constrained optimization problem becomes:

$$\min_{\boldsymbol{a}} \parallel \boldsymbol{a}\left(i+1\right) - \boldsymbol{a}\left(i\right) \parallel^2$$

$$\text{subject to}$$

$$\mid d\left(i\right) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}\left(i\right)\boldsymbol{a}\left(i+1\right) \mid = 0. \tag{33}$$

Using the method of Lagrange multipliers, we have

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

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

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

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

From equation (35) we obtain:

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

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

Substituting (36) in the equation above we get:

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

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

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

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

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

$$\boldsymbol{a}(i+1) = \begin{bmatrix} \boldsymbol{a}(i) \\ 0 \end{bmatrix} + \frac{\mu(i)}{\varepsilon + \parallel \boldsymbol{\kappa_\delta}(i) \parallel^2} e(i)\,\boldsymbol{\kappa_\delta}(i), \tag{42}$$

where $e(i) = d(i) - \boldsymbol{\kappa_\delta}^{\mathrm{T}}(i)\begin{bmatrix} \boldsymbol{a}(i) \\ 0 \end{bmatrix}$.

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

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

Substituting equation (42) in the last equation and assuming that $\frac{\|\boldsymbol{\kappa_\delta}(i)\|^2}{\varepsilon + \|\boldsymbol{\kappa_\delta}(i)\|^2} \approx 1$, we have

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

Simplifying the terms, we obtain

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

From the last equation we obtain an expression for the step-size, which is given by

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

If the error does not exceed the threshold γ, the size of the dictionary remains the same and no coefficients update is performed, only the output of the model is calculated for the new input. The pseudo-code for the NLR-SM-KNLMS algorithm is shown in Algorithm 2.

Algorithm 2 Nonlinear Regression SM-KNLMS Algorithm

Initialization

1.Choose γ , ε and κ.

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

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

4.$m = 1$

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

Computation

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

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

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

%Compute the output

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

%Compute the error

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

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

%Store the new center

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

%Store the step-size

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

%Update the coefficients

13. $\boldsymbol{a}(i+1) = \begin{bmatrix} \boldsymbol{a}(i) \\ 0 \end{bmatrix} + \frac{\mu(i)}{\varepsilon + \|\boldsymbol{\kappa_\delta}(i)\|^2} e(i)\, \boldsymbol{\kappa_\delta}(i)$

14. $m = m + 1$

15. **else**

16. $\mu(i) = 0$

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

18. **end if**

19.**end while**

V. Analysis

In this section, we consider a statistical analysis of the NLR-SM-KNLMS algorithm along with a computational complexity comparison among the proposed and existing algorithms.

A. Computational complexity

The computational complexity of the proposed algorithms and the KLMS algorithm is detailed in Table I. We consider real-valued data and the cost is given in terms of the number of multiplications and additions per iteration as a function of N, m and the update rate (UR). Moreover, the algorithms use a maximum fixed size for the dictionary, which means that the computational complexity only varies before reaching steady-state.

TABLE I

Computational Cost per update Iteration

Algorithm	Additions (+)	Multiplications (x)
KLMS	$m(N+1)+1$	$m(N+1)$
KNLMS (Regression)	$m(2N+1)+2$	$m(2N+1)+1$
C-SM-KNLMS (Algorithm 1)	$m(2N)+1+UR(1)$	$m(2N+1)+UR(1)$
NLR-SM-KNLMS (Algorithm 2)	$(m+1)(N-1)+1+UR(N+2m+1)$	$(m+1)(2N)+UR(N+m+2)$

B. Statistical Analysis

In this section, we consider a statistical analysis of the NLR-KNLMS algorithm with a Gaussian kernel in a stationary environment, which means that $\varphi(\mathbf{x}(i))$ is stationary for $\mathbf{x}(i)$ stationary [33]. We focus on the analysis of the NLR-SM-KNLMS algorithm rather than C-SM-KNLMS because the former lends itself to statistical analysis, as explained in [33].

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

The estimated output of the system is described by

$$y(i) = \boldsymbol{a}^{\mathrm{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i). \tag{47}$$

The corresponding estimation error is given by

$$e\left(i\right)=d\left(i\right)-y\left(i\right). \tag{48}$$

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

$$\begin{aligned} J_{\mathrm{ms}}\left(i\right) =&\mathbb{E}\left[e^{2}\left(i\right)\right] \\ =&\mathbb{E}\left[d^{2}\left(i\right)\right]-2\mathbf{p}_{kd}^{\mathrm{T}}\boldsymbol{a}\left(i\right)+\boldsymbol{a}^{\mathrm{T}}\left(i\right)\mathbf{R}_{kk}\boldsymbol{a}\left(i\right), \end{aligned} \tag{49}$$

where $\mathbf{R}_{kk}=\mathbb{E}\left[\boldsymbol{\kappa_{\delta}}\left(i\right)\boldsymbol{\kappa_{\delta}}^{\mathrm{T}}\left(i\right)\right]$ represents the correlation matrix of the kernelized input, and $\mathbf{p}_{kd}=\mathbb{E}\left[d\left(i\right)\boldsymbol{\kappa_{\delta}}\left(i\right)\right]$ is the cross-correlation vector between $\boldsymbol{\kappa_{\delta}}\left(i\right)$ and $d\left(i\right)$. In [34], [35] it is shown that $\mathbf{R}_{kk}$ is positive definite. Thus, the Wiener solution and the minimum MSE are obtained as follows:

$$\boldsymbol{a}_{o} = \mathbf{R}_{kk}^{-1}\mathbf{p}_{kd} \tag{50}$$

$$J_{\min} = \mathbb{E}\left[d^{2}\left(i\right)\right]-\mathbf{p}_{kd}^{\mathrm{T}}\mathbf{R}_{kk}^{-1}\mathbf{p}_{kd}, \tag{51}$$

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

$$\left[\mathbf{R}_{kk}\right]_{jl}=\begin{cases}\mathbb{E}\left[\kappa^{2}\left(\mathbf{x}\left(i\right),\mathbf{x}\left(\delta_{j}\right)\right)\right] & j=l\\ \mathbb{E}\left[\kappa\left(\mathbf{x}\left(i\right),\mathbf{x}\left(\delta_{j}\right)\right)\kappa\left(\mathbf{x}\left(i\right),\mathbf{x}\left(\delta_{l}\right)\right)\right] & j\neq l\end{cases} \tag{52}$$

Let us define the following products:

$$\left\|\mathbf{x}\left(i\right)-\mathbf{x}\left(\delta_{j}\right)\right\|^{2}=\boldsymbol{y}_{2}^{\mathrm{T}}\mathbf{Q}_{2}\boldsymbol{y}_{2} \tag{53}$$

$$\left\|\mathbf{x}\left(i\right)-\mathbf{x}\left(\delta_{j}\right)\right\|^{2}+\left\|\mathbf{x}\left(i\right)-\mathbf{x}\left(\delta_{l}\right)\right\|^{2}=\boldsymbol{y}_{3}^{\mathrm{T}}\mathbf{Q}_{3}\boldsymbol{y}_{3}, \tag{54}$$

where

$$\boldsymbol{y}_{2}=\left[\begin{array}{cc}\mathbf{x}^{\mathrm{T}}\left(i\right) & \mathbf{x}^{\mathrm{T}}\left(\delta_{j}\right)\end{array}\right]^{\mathrm{T}}, \tag{55}$$

$$\boldsymbol{y}_{3}=\left[\begin{array}{ccc}\mathbf{x}^{\mathrm{T}}\left(i\right) & \mathbf{x}^{\mathrm{T}}\left(\delta_{j}\right) & \mathbf{x}^{\mathrm{T}}\left(\delta_{l}\right)\end{array}\right]^{\mathrm{T}}, \tag{56}$$

$$\mathbf{Q}_{2}=\left[\begin{array}{cc}\mathbf{I} & -\mathbf{I}\\ -\mathbf{I} & \mathbf{I}\end{array}\right], \tag{57}$$

$$\mathbf{Q}_{3}=\left[\begin{array}{ccc}2\mathbf{I} & -\mathbf{I} & -\mathbf{I}\\ -\mathbf{I} & \mathbf{I} & \mathbf{0}\\ -\mathbf{I} & \mathbf{0} & \mathbf{I}\end{array}\right]. \tag{58}$$

We know from [36], [34] that the moment generating function of the quadratic form $z=\boldsymbol{y}^{\mathrm{T}}\mathbf{Q}\boldsymbol{y}$, where $\boldsymbol{y}$ is a zero-mean Gaussian vector with covariance matrix $\mathbf{R}_{y}$ is given by

$$\mathbb{E}\left[e^{sz}\right] = \det\left\{\mathbf{I} - 2s\mathbf{Q}\mathbf{R}_y\right\}^{-\frac{1}{2}}. \tag{59}$$

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

$$\left[\mathbf{R}_{kk}\right]_{jl} = \begin{cases} r_{md} = \det\left\{\mathbf{I}_2 - 2\mathbf{Q}_2\mathbf{R}_2/\nu^2\right\}^{-\frac{1}{2}} & j = l \\ r_{od} = \det\left\{\mathbf{I}_3 - \mathbf{Q}_3\mathbf{R}_3/\nu^2\right\}^{-\frac{1}{2}} & j \neq l \end{cases}. \tag{60}$$

Let us define the coefficients-error vector defined by

$$\boldsymbol{v}\left(i\right) = \boldsymbol{a}\left(i\right) - \boldsymbol{a}_o. \tag{61}$$

The second-order moments of the coefficients are related to the MSE through [37]

$$J_{\text{ms}}\left(i\right) = J_{\min} + tr\left\{\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\left(i\right)\right\}, \tag{62}$$

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

The update equation for the coefficients of the system is given by

$$\boldsymbol{a}\left(i+1\right) = \boldsymbol{a}\left(i\right) + \mu\left(i\right)e\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right), \tag{63}$$

where

$$\mu\left(i\right) = \begin{cases} 1 - \frac{\gamma}{|e(i)|} & |e\left(i\right)| > \gamma, \\ 0 & \text{otherwise}. \end{cases} \tag{64}$$

Subtracting $\boldsymbol{a_o}$ from equation (63), we obtain the weight error vector update equation:

$$\boldsymbol{v}\left(i+1\right) = \boldsymbol{v}\left(i\right) + \mu\left(i\right)e\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right). \tag{65}$$

The estimation error may now be rewritten as follows:

$$\begin{aligned} e\left(i\right) =& d\left(i\right) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}\left(i\right)\boldsymbol{a}\left(i\right) \\ =& d\left(i\right) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}\left(i\right)\boldsymbol{v}\left(i\right) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}\left(i\right)\boldsymbol{a}_o. \end{aligned} \tag{66}$$

The optimum error is given by

$$e_o\left(i\right) = d\left(i\right) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}\left(i\right)\boldsymbol{a}_o. \tag{67}$$

It follows that

$$e\left(i\right) = e_o\left(i\right) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}\left(i\right)\boldsymbol{v}\left(i\right). \tag{68}$$

We may represent equation (63) by

$$\boldsymbol{a}\left(i+1\right)=\boldsymbol{a}\left(i\right)+P_{\text{up}}\left(1-\frac{\gamma}{\left|e\left(i\right)\right|}\right)e\left(i\right)\boldsymbol{\kappa_{\delta}}\left(i\right), \tag{69}$$

where $P_{\text{up}}=\Pr(|e(i)|>\gamma)=2Q\left(\frac{\gamma}{\sigma_e}\right)$ denotes the probability of update of the set-membership algorithm [28] and σ_e is the standard deviation of a Gaussian random variable associated with the error.

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

$$\begin{aligned}\boldsymbol{v}\left(i+1\right)=&\boldsymbol{v}\left(i\right)+P_{\text{up}}\left(1-\frac{\gamma}{\left|e\left(i\right)\right|}\right)e\left(i\right)\boldsymbol{\kappa_{\delta}}\left(i\right)\\ =&\boldsymbol{v}\left(i\right)+P_{\text{up}}e\left(i\right)\boldsymbol{\kappa_{\delta}}\left(i\right)\\ &-\gamma P_{\text{up}}\text{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa_{\delta}}\left(i\right),\end{aligned} \tag{70}$$

Replacing (68) in the equation above we obtain

$$\begin{aligned}\boldsymbol{v}\left(i+1\right)=&\boldsymbol{v}\left(i\right)+P_{\text{up}}\left(e_o\left(i\right)-\boldsymbol{\kappa_{\delta}}^{\text{T}}\left(i\right)\boldsymbol{v}\left(i\right)\right)\boldsymbol{\kappa_{\delta}}\left(i\right)\\ &-\gamma P_{\text{up}}\text{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa_{\delta}}\left(i\right)\\ =&\boldsymbol{v}\left(i\right)+P_{\text{up}}e_o\left(i\right)\boldsymbol{\kappa_{\delta}}\left(i\right)\\ &-P_{\text{up}}\boldsymbol{\kappa_{\delta}}\left(i\right)\boldsymbol{\kappa_{\delta}}^{\text{T}}\left(i\right)\boldsymbol{v}\left(i\right)\\ &-\gamma P_{\text{up}}\text{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa_{\delta}}\left(i\right).\end{aligned} \tag{71}$$

Post-multiplying equation (71) by its transpose and taking the expected value leads to:

$$
\begin{aligned}
\mathbf{C}_{\boldsymbol{v}}(i+1) =& \mathbf{C}_{\boldsymbol{v}}(i) + P_{\text{up}}\mathbb{E}\left[e_o(i)\,\boldsymbol{v}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
&- P_{\text{up}}\mathbb{E}\left[\boldsymbol{v}(i)\,\boldsymbol{v}^{\text{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
&- P_{\text{up}}\gamma\mathbb{E}\left[\text{sgn}(e(i))\,\boldsymbol{v}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
&+ P_{\text{up}}\mathbb{E}\left[e_o(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{v}^{\text{T}}(i)\right] \\
&+ P_{\text{up}}^2\mathbb{E}\left[e_o^2(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
&- P_{\text{up}}^2\mathbb{E}\left[e_o(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{v}^{\text{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
&- 2P_{\text{up}}^2\gamma\mathbb{E}\left[e_o(i)\,\text{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
&- P_{\text{up}}\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{v}^{\text{T}}(i)\right] \\
&- P_{\text{up}}^2\mathbb{E}\left[e_o(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
&+ P_{\text{up}}^2\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{v}^{\text{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
&+ P_{\text{up}}^2\gamma\mathbb{E}\left[\text{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
&- P_{\text{up}}\gamma\mathbb{E}\left[\text{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{v}^{\text{T}}(i)\right] \\
&+ P_{\text{up}}^2\gamma\mathbb{E}\left[\text{sgn}(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{v}^{\text{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
&+ P_{\text{up}}^2\gamma^2\mathbb{E}\left[\text{sgn}^2(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right]. \qquad (72)
\end{aligned}
$$

Let us define $\mathbf{T}(i) = \mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{v}^{\text{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right]$ to simplify the notation. Assuming that the inputs and the coefficients are statistically independent, then the following expected values are reduced to

$$
\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\,\boldsymbol{v}(i)\,\boldsymbol{v}^{\text{T}}(i)\right] = \mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}(i), \qquad (73)
$$

$$
\mathbb{E}\left[\boldsymbol{v}(i)\,\boldsymbol{v}^{\text{T}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] = \mathbf{C}_{\boldsymbol{v}}(i)\,\mathbf{R}_{kk}. \qquad (74)
$$

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

$$
\begin{aligned}
\mathbb{E}\left[e_o^2(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \approx& \mathbb{E}\left[e_o^2(i)\right]\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
\approx& J_{\min}\mathbf{R}_{kk}, \qquad (75)
\end{aligned}
$$

and

$$
\begin{aligned}
\mathbb{E}\left[\text{sgn}^2(e(i))\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \approx& \mathbb{E}\left[\text{sgn}^2(e(i))\right]\mathbb{E}\left[\boldsymbol{\kappa}_{\boldsymbol{\delta}}(i)\,\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\text{T}}(i)\right] \\
\approx& \mathbf{R}_{kk}. \qquad (76)
\end{aligned}
$$

By the orthogonality principle, we obtain:

$$\begin{aligned}\mathbb{E}\left[e_o\left(i\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \approx & \mathbb{E}\left[e_o\left(i\right)\boldsymbol{\kappa_\delta}\left(i\right)\right]\mathbb{E}\left[\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \\ \approx & \mathbf{0},\end{aligned} \tag{77}$$

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

$$\begin{aligned}&\mathbb{E}\left[e_o\left(i\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &= \mathbb{E}\left[\boldsymbol{v}^{\mathrm{T}}\left(i\right)e_o\left(i\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\approx \mathbb{E}\left[\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right]\mathbb{E}\left[e_o\left(i\right)\boldsymbol{\kappa_\delta}\left(i\right)\right]\mathbb{E}\left[\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\approx \mathbf{0}.\end{aligned} \tag{78}$$

With the results of equations (73),(74), (75), (76), (77) and (78), equation (72) is reduced to:

$$\begin{aligned}\mathbf{C}_{\boldsymbol{v}}\left(i+1\right) = & \mathbf{C}_{\boldsymbol{v}}\left(i\right) - P_{\mathrm{up}}\mathbf{C}_{\boldsymbol{v}}\left(i\right)\mathbf{R}_{kk} \\ & - P_{\mathrm{up}}\gamma\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ & + P_{\mathrm{up}}^2 J_{\min}\mathbf{R}_{kk} \\ & - 2P_{\mathrm{up}}^2\gamma\mathbb{E}\left[e_o\left(i\right)\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ & - P_{\mathrm{up}}\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\left(i\right) + P_{\mathrm{up}}^2\mathbf{T}\left(i\right) \\ & + P_{\mathrm{up}}^2\gamma\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ & - P_{\mathrm{up}}\gamma\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \\ & + P_{\mathrm{up}}^2\gamma\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ & + P_{\mathrm{up}}^2\gamma^2\mathbf{R}_{kk},\end{aligned} \tag{79}$$

The remaining expected values of (79) can be computed using Price's theorem [38]. For the ninth term, the expected value may be approximated as follows:

$$\begin{aligned}&\mathbb{E}\left[\mathrm{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \\ &\approx \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e\left(i\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \\ &\approx \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[\left(e_o\left(i\right) - \boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\right)\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \\ &\approx -\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[\boldsymbol{\kappa_\delta}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{v}^{\mathrm{T}}\left(i\right)\right] \\ &\approx -\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\left(i\right).\end{aligned} \tag{80}$$

Calculating the third term of equation (79), we obtain

$$\begin{aligned}\mathbb{E}\left[\operatorname{sgn}\left(e\left(i\right)\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \approx & \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ \approx & -\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{C}_{\boldsymbol{v}}\left(i\right)\mathbf{R}_{kk}. \end{aligned} \tag{81}$$

The sixth term of equation (79) is given by

$$\begin{aligned}&\mathbb{E}\left[e_o\left(i\right)\operatorname{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\quad\approx \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e_o\left(i\right)e\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\quad\approx \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e_o^2\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\qquad-\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e_o\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\quad\approx \sqrt{\frac{2}{\pi\sigma_e^2}}J_{\min}\mathbf{R}_{kk}. \end{aligned} \tag{82}$$

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

$$\begin{aligned}&\mathbb{E}\left[\operatorname{sgn}\left(e\left(i\right)\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\quad\approx \sqrt{\frac{2}{\pi\sigma_e^2}}\mathbb{E}\left[e\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\boldsymbol{v}\left(i\right)\boldsymbol{\kappa}_{\boldsymbol{\delta}}^{\mathrm{T}}\left(i\right)\right] \\ &\quad\approx -\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{T}\left(i\right). \end{aligned} \tag{83}$$

The results obtained in (80), (81), (82) and (83) shall turn (79) into:

$$\begin{aligned}\mathbf{C}_{\boldsymbol{v}}\left(i+1\right) = & \mathbf{C}_{\boldsymbol{v}}\left(i\right) - P_{\mathrm{up}}\mathbf{C}_{\boldsymbol{v}}\left(i\right)\mathbf{R}_{kk} \\ & + P_{\mathrm{up}}\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{C}_{\boldsymbol{v}}\left(i\right)\mathbf{R}_{kk} \\ & + P_{\mathrm{up}}^2 J_{\min}\mathbf{R}_{kk} - 2P_{\mathrm{up}}^2\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}J_{\min}\mathbf{R}_{kk} \\ & - P_{\mathrm{up}}\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\left(i\right) + P_{\mathrm{up}}^2\mathbf{T}\left(i\right) \\ & - 2P_{\mathrm{up}}^2\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{T}\left(i\right) \\ & + P_{\mathrm{up}}\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\left(i\right) \\ & + P_{\mathrm{up}}^2\gamma^2\mathbf{R}_{kk}. \end{aligned} \tag{84}$$

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

$$\begin{aligned}\mathbf{C}_{\boldsymbol{v}}(i+1) =& \mathbf{C}_{\boldsymbol{v}}(i) + P_{\text{up}}^2\left(1-2\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}\right)\left(J_{\min}\mathbf{R}_{kk}+\mathbf{T}(i)\right)\\ &+ P_{\text{up}}\left(\gamma\sqrt{\frac{2}{\pi\sigma_e^2}}-1\right)\left(\mathbf{C}_{\boldsymbol{v}}(i)\mathbf{R}_{kk}+\mathbf{R}_{kk}\mathbf{C}_{\boldsymbol{v}}\right)\\ &+ P_{\text{up}}^2\gamma^2\mathbf{R}_{kk}. \end{aligned} \tag{85}$$

The authors of [34] proved that the elements of $\mathbf{T}(i)$ are given by

$$\begin{aligned}\left[\mathbf{T}(i)\right]_{jj} =& \sum_{\substack{l=1\\ l\neq j}}^{M}\left\{2\mu_2\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{jl}+\mu_3\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{ll}+\mu_4\sum_{\substack{p=1\\ p\neq\{j,l\}}}^{M}\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{lp}\right\}\\ &+\mu_1\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{jj}, \end{aligned} \tag{86}$$

for the main diagonal elements and

$$\begin{aligned}\left[\mathbf{T}(i)\right]_{jk} =& \mu_2\left(\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{jj}+\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{kk}\right)+2\mu_3\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{jk}\\ &+\sum_{\substack{l=1\\ l\neq\{j,k\}}}^{M}\left\{2\mu_4\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{kl}+2\mu_4\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{jl}+\mu_4\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{ll}\right.\\ &\left.+\mu_5\sum_{\substack{p=1\\ p\neq\{j,k,l\}}}^{M}\left[\mathbf{C}_{\boldsymbol{v}}(i)\right]_{lp}\right\}, \end{aligned} \tag{87}$$

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

$$\mu_1 = \det\left\{\mathbf{I}_2-4\mathbf{Q}_2\mathbf{R}_2/\nu^2\right\}^{-\frac{1}{2}}, \tag{88}$$

$$\mu_2 = \det\left\{\mathbf{I}_3-\mathbf{Q}_{3'}\mathbf{R}_3/\nu^2\right\}^{-\frac{1}{2}}, \tag{89}$$

$$\mu_3 = \det\left\{\mathbf{I}_3-2\mathbf{Q}_{3'}\mathbf{R}_3/\nu^2\right\}^{-\frac{1}{2}}, \tag{90}$$

$$\mu_4 = \det\left\{\mathbf{I}_4-2\mathbf{Q}_4\mathbf{R}_4/\nu^2\right\}^{-\frac{1}{2}}, \tag{91}$$

$$\mu_5 = \det\left\{\mathbf{I}_5-2\mathbf{Q}_5\mathbf{R}_5/\nu^2\right\}^{-\frac{1}{2}}, \tag{92}$$

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

$$\mathbf{Q}_{3'} = \begin{bmatrix} 4\mathbf{I} & -3\mathbf{I} & -\mathbf{I}\\ -3\mathbf{I} & 3\mathbf{I} & \mathbf{0}\\ -\mathbf{I} & \mathbf{0} & \mathbf{I}\end{bmatrix}, \tag{93}$$

$$\mathbf{Q}_4 = \begin{bmatrix} 4\mathbf{I} & -2\mathbf{I} & -\mathbf{I} & -\mathbf{I} \\ -2\mathbf{I} & 2\mathbf{I} & \mathbf{0} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}, \tag{94}$$

$$\mathbf{Q}_5 = \begin{bmatrix} 4\mathbf{I} & -\mathbf{I} & -\mathbf{I} & -\mathbf{I} & -\mathbf{I} \\ -\mathbf{I} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix}. \tag{95}$$

Replacing equations (86) and (87) into equation (85) leads us to a recursive expression for the entries of the autocorrelation matrix $\mathbf{C}_{\boldsymbol{v}}(i)$:

$$\begin{aligned} [\mathbf{C}_{\boldsymbol{v}}(i+1)]_{jj} = {} & \left(1 + 2P_{\mathrm{up}} a r_{md} + P_{\mathrm{up}}^2 b \mu_1\right) [\mathbf{C}_{\boldsymbol{v}}(i)]_{jj} \\ & + P_{up}^2 b \mu_3 \sum_{\substack{l=1 \\ l \neq j}}^{M} [\mathbf{C}_{\boldsymbol{v}}(i)]_{ll} \\ & + \left(2P_{\mathrm{up}}^2 \mu_2 b + 2P_{\mathrm{up}} a r_{od}\right) \sum_{\substack{l=1 \\ l \neq j}}^{M} [\mathbf{C}_{\boldsymbol{v}}(i)]_{jl} \\ & + P_{\mathrm{up}}^2 \mu_2 b \mu_4 \sum_{\substack{l=1 \\ l \neq j}}^{M} \sum_{\substack{p=1 \\ p \neq \{j,l\}}}^{M} [\mathbf{C}_{\boldsymbol{v}}(i)]_{lp} \\ & + \left(P_{\mathrm{up}}^2 b J_{\min} + P_{\mathrm{up}}^2 \gamma^2\right) r_{md}, \end{aligned} \tag{96}$$

and for $j \neq k$

$$\begin{aligned}
\left[\mathbf{C}_{\boldsymbol{v}}\left(i+1\right)\right]_{jk} = & \left(1+2P_{\mathrm{up}}\alpha r_{md}+2P_{\mathrm{up}}^{2}\beta\mu_{3}\right)\left[\mathbf{C}_{\boldsymbol{v}}\left(i\right)\right]_{jk} \\
& + P_{\mathrm{up}}^{2}\beta\mu_{4}\sum_{\substack{l=1 \\ l\neq\{j,k\}}}^{M}\left[\mathbf{C}_{\boldsymbol{v}}\left(i\right)\right]_{ll}+\Bigg(P_{\mathrm{up}}^{2}\beta\mu_{2} \\
& + P_{\mathrm{up}}\alpha r_{od}\Bigg)\left(\left[\mathbf{C}_{\boldsymbol{v}}\left(i\right)\right]_{jj}+\left[\mathbf{C}_{\boldsymbol{v}}\left(i\right)\right]_{kk}\right) \\
& + \left(2P_{\mathrm{up}}^{2}\beta\mu_{4}+P_{\mathrm{up}}\alpha r_{od}\right)\sum_{\substack{l=1 \\ l\neq\{j,k\}}}^{M}\Bigg(\left[\mathbf{C}_{\boldsymbol{v}}\left(i\right)\right]_{il} \\
& + \left[\mathbf{C}_{\boldsymbol{v}}\left(i\right)\right]_{jl}\Bigg)+P_{\mathrm{up}}^{2}\beta\mu_{5}\sum_{\substack{l=1 \\ l\neq\{j,k\}}}^{M}\sum_{\substack{p=1 \\ p\neq\{j,k,l\}}}^{M}\left[\mathbf{C}_{\boldsymbol{v}}\left(i\right)\right]_{lp} \\
& + \left(P_{\mathrm{up}}^{2}\beta J_{\min}+P_{\mathrm{up}}^{2}\gamma^{2}\right)r_{md}, \qquad (97)
\end{aligned}$$

where

$$\alpha = \gamma\sqrt{\frac{2}{\pi\sigma_{e}^{2}}}-1, \qquad (98)$$

$$\beta = 1-2\gamma\sqrt{\frac{2}{\pi\sigma_{e}^{2}}}. \qquad (99)$$

The entries of the autocorrelation matrix $\mathbf{C}_{\boldsymbol{v}}\left(i\right)$ are then plugged in the MSE expression in (62).

VI. Simulations

In this section we assess the performance of the SM-KNLMS algorithms proposed. The Gaussian kernel was used in all the algorithms to perform all the experiments. We have structured this section into two parts: the first part deals with the identification of nonlinear systems, whereas the second part examines time series prediction problems.

A. System identification

In the first example, we consider a system identification application to compare the performance of the proposed SM-KNLMS algorithms and to verify the theory developed in Section V. Let us consider the nonlinear problem studied in [39], [40], [34] described by the recursion

$$d\left(i\right) = \frac{d\left(i-1\right)}{1+d\left(i-1\right)}+x^{3}\left(i-1\right). \qquad (100)$$

We compare the performance of the proposed algorithms with the KLMS algorithm. The desired signal $d\left(i\right)$ was corrupted by additive white Gaussian noise with zero-mean and variance $\sigma_{n}^{2}=10^{-4}$ and the SNR was set to 20 dB. We have also considered a fixed dictionary of length of 16 in order to focus solely on the performance of the

gradient learning rules used by the analyzed algorithms. At each iteration, the dictionary elements were updated so that the oldest element added is replaced. To compute the learning curve, a total of 500 simulations were averaged, each one with 1500 iterations. The bandwidth of the Gaussian kernel was set to 0.025. The threshold for the SM-KNLMS algorithms was set to $\gamma = \sqrt{5}\sigma_n$. The result of this experiment is presented in Fig. 1. The results show that the C-SM-KNLMS algorithm slightly outperforms in learning rate the NLR-SM-KNLMS algorithm and the nonlinear regression-based KLMS (NLR-KLMS) algorithm. At steady state C-SM-KNLMS and NLR-KNLMS tend to produce comparable results, which means that the analytical formulas to predict the results of NLR-KNLMS can be useful to have a prediction of the performance of C-SM-NKLMS at steady state. For this reason we will consider the C-SM-KNLMS algorithm for most examples except for those that show analytical results and employ the NLR-SM-KNLMS algorithm, which is the only one suitable for statistical analysis.

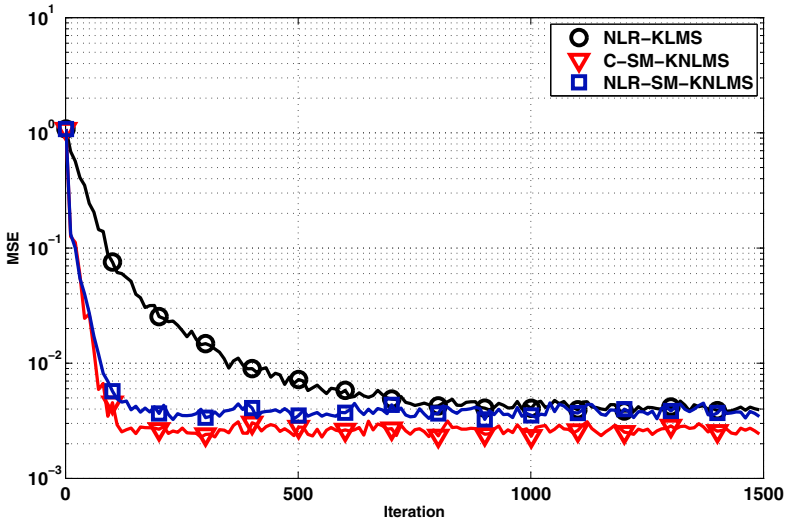

Fig. 1. Performance comparison of SM-KNLMS algorithms.

In the second example we evaluate the transient behavior of the NLR-SM-KNLMS. The input sequence $x\,(i)$ is independent and identically Gaussian distributed with variance $\sigma_x^2 = 0.15$. We have also considered a fixed dictionary of length 16. At each iteration, the dictionary elements were updated so that the oldest element added is replaced. To compute the learning curve, a total of 500 simulations were averaged, each one with 3000 iterations. The bandwidth of the Gaussian kernel was set to 0.025 for Fig. 2 and the threshold was set to $\gamma = \sqrt{10}\sigma_n$ to obtain the results in Fig. 2.

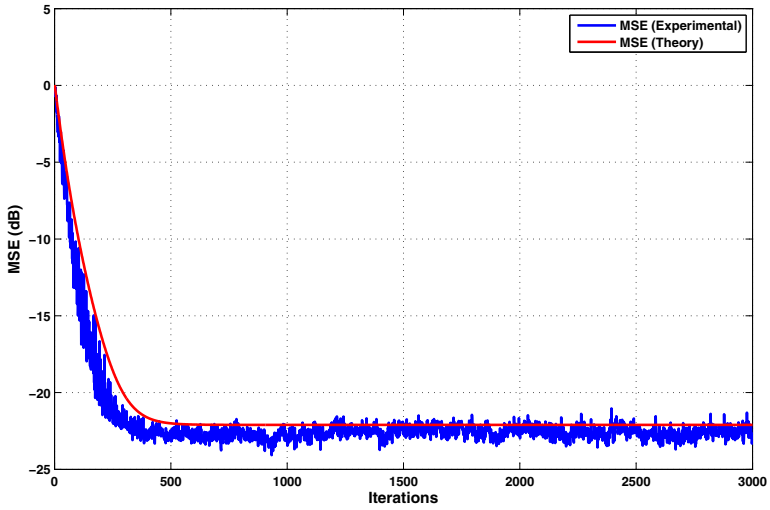

Fig. 2. Transient behaviour of the SM-KNLMS algorithm. Bandwidth=0.025

In the third example, we assess the performance of the NLR-SM-KNLMS algorithm in a non-stationary environment. Particularly, we investigate the case when a sudden change occurs the system model, resulting in a different value of $\boldsymbol{\alpha_o}$. The two systems studied are given by

$$d_1(i) = \frac{d(i-1)}{1+d(i-1)} + x^3(i-1), \tag{101}$$

$$d_2(i) = x^2(i). \tag{102}$$

In particular, a total of 8000 iterations were made, where the first 4000 iterations correspond to system d_1. Then, the system becomes unstable for 50 iterations where $d(i) = d(i-1) + 0.1$. The remaining iterations correspond to system d_2. The output is corrupted by AWGN with standard deviation equal to $\sigma_n = 0.01$. The input follows a Gaussian distribution with i.i.d samples and standard deviation given by $\sigma_x = 0.15$.

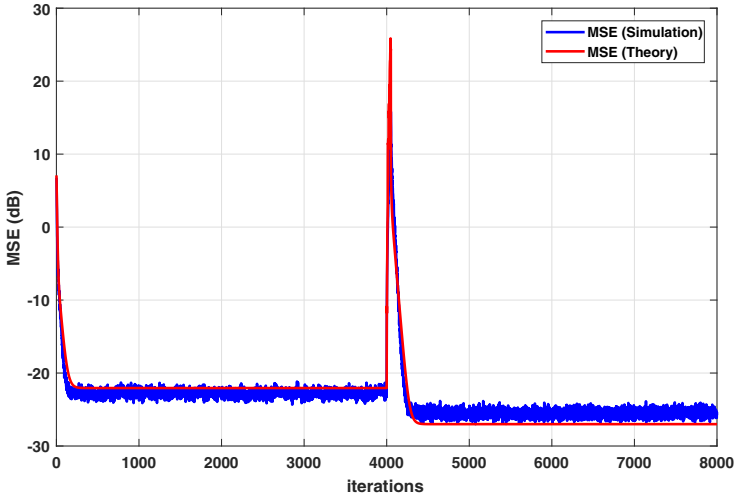

Fig. 3. Performance of the NLR-SM-KNLMS algorithm in a non-stationary environment.

From Fig. 3 we note that NLR-SM-KNLMS is capable of tracking changes on the system and of converging to a new solution in few iterations. The solution obtained for system d_2 achieves a lower MSE because the correlation between the mapped input and the desired signal is higher for this system. It is also important to mention that the simulation result matches the theoretical result.

In the fourth experiment we assess the performance of NLR-SM-KNLMS for correlated inputs. Let us consider two inputs, $\mathbf{x}_{c_1}$ and $\mathbf{x}_{c_2}$ each one with three different components i.e. $\mathbf{x}_c(i) = \begin{bmatrix} x_{c,1}(i) & x_{c,2}(i) & x_{c,3}(i) \end{bmatrix}^T$. The correlation of the inputs satisfies

$$x_{c_1,2}(i) = 0.5x_{c_1,1}(i) + \delta_x(i) \tag{103}$$

$$x_{c_2,2}(i) = 0.5x_{c_2,1}(i) + \delta_x(i) \tag{104}$$

$$x_{c_2,3}(i) = 0.2x_{c_2,1}(i) + 0.4x_{c_2,2}(i) + \delta_x(i) \tag{105}$$

Both signals pass through a linear system with memory where the output is given by

$$y(i) = \boldsymbol{r}^T\mathbf{x}_c(i) - 0.3y(i-1) + 0.35y(i-2) \tag{106}$$

with $\boldsymbol{r} = \begin{bmatrix} 1 & 0.5 & 0.3 \end{bmatrix}$. A nonlinear function is then applied to $y(i)$

$$d(i) \begin{cases} \frac{y(i)}{3(0.1+0.9y^2(i))^{1/2}} & y(i) \geq 0 \\ \frac{-y^2(i)\left[1-e^{0.7y(i)}\right]}{3} & y(i) < 0 \end{cases} \tag{107}$$

The desired signal is corrupted by AWGN with $\sigma_n = 0.001$. Fig. 4 illustrates the performance of NLR-KNLMS. The results show that the convergence speed for both inputs is similar. However, the correlation between the elements of the second input is stronger than that of the first input. This affects directly the performance of the

NLR-SM-KNLMS as shown in Fig. 4, where we can see that the first input achieves a lower MSE than the second input.

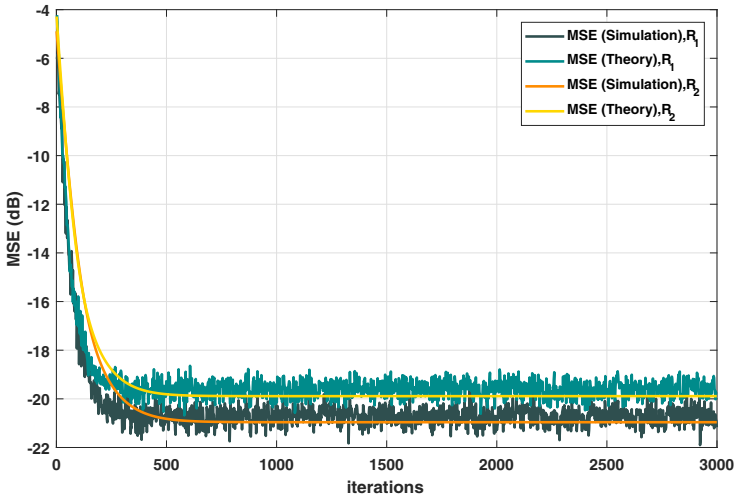

Fig. 4. Performance of the NLR-SM-KNLMS algorithm with correlated inputs.

In the last experiment of this section, we consider the identification of a Hammerstein system [41]. The input vector $\mathbf{x}(i) \in \mathbb{R}^{1\times 24}$ where each element has $\sigma_x^2 = 4 \times 10^{-4}$ and the noise variance is $\sigma_n = 10^{-6}$. The kernel bandwidth was set to 0.048. The input goes through a nonlinear function to form the vector $\tilde{\mathbf{x}}(i)$, where each element is given by

$$\tilde{x}_j(i) = x_j^3(i) \tag{108}$$

The desired signal is obtained from a linear system expressed by

$$d(n) = \mathbf{s}^T \tilde{\mathbf{x}}(i) \tag{109}$$

with $s_1 = 1$, $s_2 = 0.5$, $s_3 = 0.3$, $s_4 = s_5 = s_9 = s_{13} = s_{15} = s_{19} = s_{22} = 0.1$, $s_6 = s_7 = -0.2$, $s_8 = s_{10} = s_{14} = -0.15$, $s_{18} = 0.15$, $s_9 =, s_{11} = 0.12$, $s_{12} = -0.09$, $s_{16} = 0.05$, $s_{17} = -0.05$, $s_{20} = 0.03$, $s_{21} = -0.12$, $s_{23} = -0.02$, $s_{24} = -0.01$.

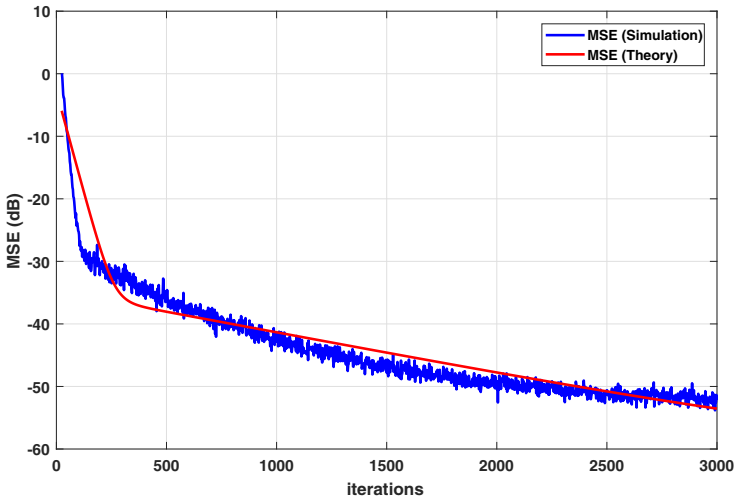

Fig. 5. Performance of the NLR-SM-KNLMS algorithm for a Hammerstein system.

The results shown in Fig. 5 indicate that the learning speed of the proposed NLR-SM-KNLMS is lower than that for the identification of other nonlinear systems. This is because the Hammerstein system considered here has a larger number of parameters, which requires more iteration for the identification. The curves in Fig. 5 also show that the theoretical results agree well with those obtained by simulations.

B. Time series prediction

Let us now consider the performance of the proposed algorithms for a time series prediction task. We have used two different time series to perform the tests, the Mackey Glass time series [42] and a laser generated time series. First, we separate the data into two sets, one for training and the other for testing as suggested in [1]. The time-window was set to seven and the prediction step was set to one so that the last seven inputs of the time series were used to predict the value one step ahead. Additionally, both time series were corrupted by additive Gaussian noise with zero mean and standard deviation equal to 0.04. Using the Silverman rule and after several tests, the bandwidth of the kernel was optimized and the optimum value found was one.

First we evaluate the performance of the adaptive algorithms over the Mackey-Glass time series, which is generated by a nonlinear time difference equation that can be used to model nonlinear dynamics including chaos and represents a challenging time series for prediction tasks [42]. A total of 1500 sample inputs were used to generate the learning curve and the prediction was performed over 100 test samples. For the KLMS algorithm the step size was set to 0.05. The error bound for the C-SM-KNLMS algorithm was set to $\sqrt{5}\sigma$. The final results of the algorithms tested are shown in Table II where the last 100 data points of each learning curve were averaged to obtain the MSE. The learning curves of the algorithms based on kernels is presented in Fig. 6. From the curves, we see that the proposed C-SM-KNLMS algorithm outperforms conventional algorithms in convergence speed.

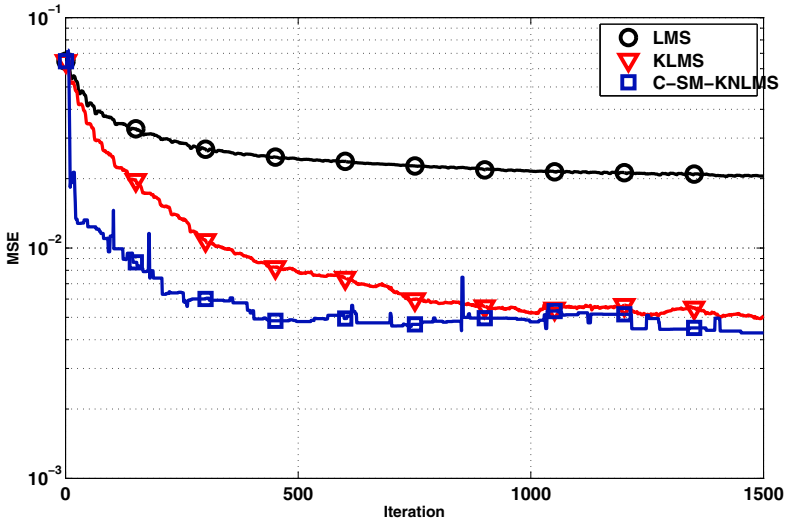

Fig. 6. Learning Curve of the Kernel Adaptive Algorithms for the Mackey-Glass Time Series prediction

TABLE II

PERFORMANCE ON MACKEY-GLASS TIME SERIES PREDICTION

Algorithm	Test MSE	Standard Deviation
LMS	0.023	+/-0.0002
NLMS	0.021	+/-0.0001
SM-NLMS	0.020	+/-0.0008
KLMS	0.007	+/-0.0003
C-SM-KNLMS	0.005	+/-0.0004

In the second example of this section, we consider the performance of the proposed algorithms over a laser generated time series, which is generated by chaotic intensity pulsations of a laser and also represents a challenging time series for prediction tasks [1]. In this case, 3500 sample inputs were used to generate the learning curves and the prediction was performed over 100 test samples. The setup used in the previous experiment was considered. Table III summarizes the MSE obtained for every algorithm tested. The learning curves are shown in Fig. 7.

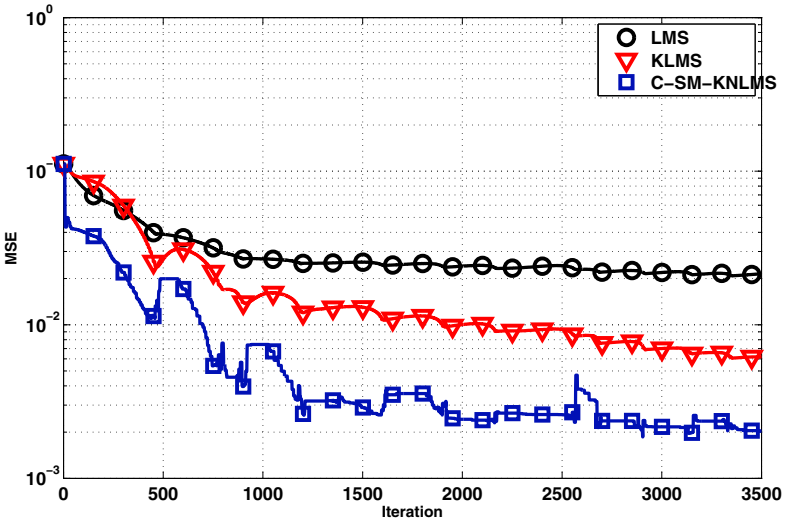

Fig. 7. Learning curves for the Laser Time Series prediction

TABLE III

PERFORMANCE ON LASER GENERATED TIME SERIES PREDICTION

Algorithm	Test MSE	Standard Deviation
LMS	0.021	+/-0.0003
NLMS	0.019	+/-0.001
SM-NLMS	0.024	+/-0.006
KLMS	0.009	+/-0.0006
C-SM-KNLMS	0.003	+/-0.0005

In the third experiment of this section we study the size of the dictionary generated by the conventional KLMS algorithm using different criteria to limit the size and by the proposed C-SM-KNLMS algorithm. The result is presented in Fig. 8. We notice that the proposed C-SM-KNLMS algorithm naturally limits the size of the dictionary. We also compare the performance of the C-SM-KNLMS with the performance obtained by the KLMS algorithm with different criteria. Fig. 9 summarizes the results, which shows that the proposed C-SM-KNLMS algorithm outperforms the existing algorithms by a significant margin.

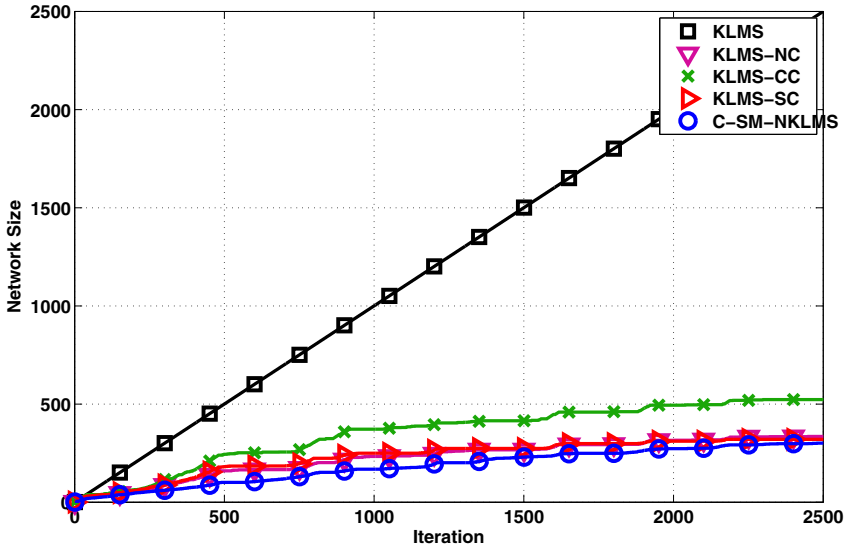

Fig. 8. Dictionary Size vs Iterations

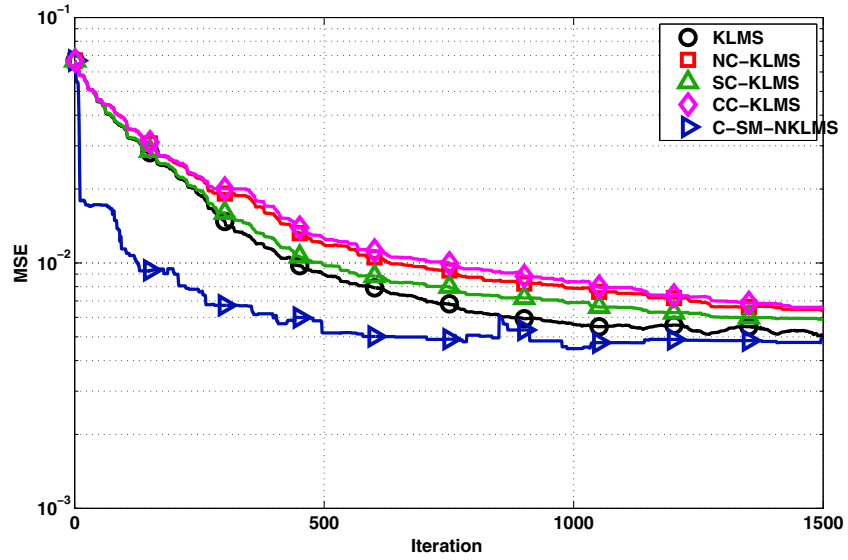

Fig. 9. Performance comparison C-SM-KNLMS vs KLMS over time iterations.

In the last experiment, we have assessed the robustness of the proposed and existing algorithms for Gaussian noise with different values of standard deviation. Fig. 10 shows the results in terms of MSE performance against the noise standard deviation. The curves obtained in Fig. 10 indicate that the proposed C-SM-KNLMS algorithm outperforms the other algorithms for all the range of values of noise standard deviation considered. As expected the performance of all algorithms evaluated gradually degrade as the noise standard deviation increases.

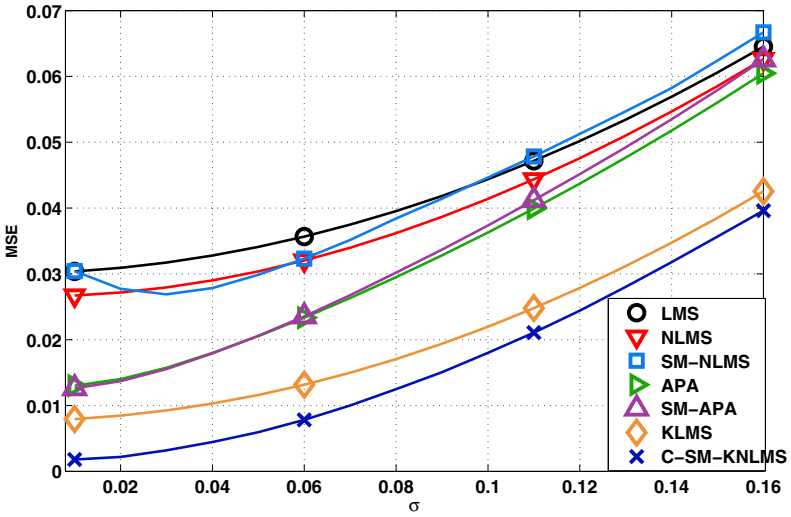

Fig. 10. Robustness performance of the studied algorithms versus standard deviation of noise.

VII. Conclusions

In this paper, we have devised data-selective kernel-type algorithms, namely, the centroid-based and the nonlinear regression SM-KNLMS algorithms. The proposed SM-KNLMS algorithms have a faster convergence speed and a lower computational cost than the existing kernel-type algorithms in the same category. The proposed SM-KNLMS algorithms also have the advantage of naturally limiting the size of the dictionary created by kernel based algorithms and a satisfactory noise robustness. These features allow the proposed SM-KNLMS algorithms to solve nonlinear filtering and estimation problems with a large number of parameters without requiring a much longer training or computational cost. Simulations have shown that the proposed SM-KNLMS algorithms outperform previously reported techniques in examples of nonlinear system identification and prediction of a time series originating from a nonlinear difference equation.

Acknowledgment

The authors would like to thank the CNPq, and FAPERJ Brazilian agencies for funding.

References

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

[2] J. M. Gil-Cacho, M. Signoretto, T. van Waterschoot, M. Moonen, and S. Jensen, "Nonlinear acoustic echo cancellation based on a sliding-window leaky kernel affine projection algorithm." *IEEE Transactions on Audio, Speech and Language Processing*, vol. 21, no. 9, pp. 1867 – 1878, April 2013.

[3] J. M. Gil-Cacho, T. van Waterschoot, M. Moonen, and S. Jensen, "Nonlinear acoustic echo cancellation based on a parallel-cascade kernel affine projection algorithm." *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012.

[4] Y. Nakijama and M. Yukawa, "Nonlinear channel equalization by multi-kernel adaptive filter." *IEEE 13th International Workshop on Signal Processing Advances in Wireless Communications*, 2012.

[5] C. Richard, J. Bermudez, and P. Honeine, "Online prediction of time series data with kernels." *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, Feb. 2009.

[6] W. Liu, P. Pokharel, and J. Príncipe, "The kernel least-mean-squares algorithm." *IEEE Transactions on Signal Processing*, vol. 56, no. 2, pp. 543–554, February 2008.

[7] P. Boboulis 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, pp. 964–978, March 2011.

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

[9] K. Slavakis and S. Theodoridis, "Sliding window generalized kernel affine projection algorithm using projection mappings," *EURASIP Journal on Advances in Signal Processing*, vol. 2008, no. 1, p. 735351, Apr 2008. [Online]. Available: https://doi.org/10.1155/2008/735351

[10] K. Slavakis, S. Theodoridis, and I. Yamada, "Online kernel-based classification using adaptive projection algorithms," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 2781–2796, July 2008.

[11] S. Theodoridis, K. Slavakis, and I. Yamada, "Adaptive learning in a world of projections," *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 97–123, Jan 2011.

[12] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm." *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, August 2004.

[13] W. Liu, Y. Wang, and J. Príncipe, "Extended kernel recursive least squares algorithm." *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3801–3814, May 2009.

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

[15] M. Yukawa, "Multikernel adaptive filtering." *IEEE Transactions on Signal Processing*, vol. 60, no. 9, pp. 4672 – 4682, August 2012.

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

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

[18] F. Sheikholeslami, D. Berberidis, and G. B. Giannakis, "Kernel-based low-rank feature extraction on a budget for big data streams." *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2015.

[19] J. Platt, "A resource-allocating network for function interpolation." *Neural Computation*, vol. 3, no. 3, pp. 213–225, 1991.

[20] W. Liu and J. Príncipe, "An information theoretic approach of designing sparse kernel adaptive filters." *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1950 – 1961, November 2009.

[21] A. Flores and R. C. de Lamare, "Set-membership kernel adaptive algorithms," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 2676–2680.

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

[23] S. Gollamudi, S. Nagaraj, S. Kapoor, and Y. F. Huang, "Set-membership filtering and a set-membership normalized LMS algorithm with an adaptive step size." *IEEE Signal Processing Letters*, vol. 5, no. 5, pp. 111–114, May 1998.

[24] S. Werner and P. Diniz, "Set-membership affine projection algorithm." *IEEE Signal Processing Letters*, vol. 8, no. 8, pp. 231–235, August 2001.

[25] P. Diniz and S. Werner, "Set-membership binormalized data-reusing LMS algorithms." *IEEE Transactions on Signal Processing*, vol. 51, no. 1, pp. 124–134, January 2003.

[26] R. C. de Lamare and P. Diniz, "Set-membership adaptive algorithms based on time-varying error bounds for CDMA interference suppression." *IEEE Transactions on Vehicular Technology*, vol. 58, no. 2, pp. 644 – 654, February 2009.

[27] T. Wang, R. C. de Lamare, and P. D. Mitchell, "Low-complexity set-membership channel estimation for cooperative wireless sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 6, pp. 2594–2607, July 2011.

[28] R. C. de Lamare and P. Diniz, "Blind adaptive interference suppression based on set-membership constrained constant-modulus algorithms with dynamic bounds." *IEEE Transactions on Signal Processing*, vol. 61, no. 5, pp. 1288 – 1301, November 2012.

[29] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. a. Scholkopf, "An introduction to kernel-based learning algorithms." *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 181 – 201, March 2001.

[30] R. C. de Lamare and R. Sampaio-Neto, "Adaptive reduced-rank processing based on joint and iterative interpolation, decimation, and filtering," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2503–2514, July 2009.

[31] R. Coelho, V. H. Nascimento, R. Queiroz, J. Romano, and C. Cavalcante, Eds., *Signals and Images: Advances and Results in Speech, Estimation, Compression, Recognition, Filtering, and Processing.* CRC Press, 2015.

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

[33] J. Chen, W. Gao, C. Richard, and J. C. Bermudez, "Convergence analysis of kernel LMS algorithm with pre-tuned dictionary." *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014.

[34] W. Parreira, J. C. Bermudez, C. Richard, and J. Tourneret, "Stochastic behavior analysis of the Gaussian kernel least-mean-square algorithm." *IEEE Transactions on Signal Processing*, vol. 60, no. 5, pp. 2208 – 2222, January 2012.

[35] ——, "Steady-state behavior and design of the Gaussian KLMS algorithm." *European Signal Processing Conference (EUSIPCO)*, April 2011.

[36] J. Omura and T. Kailath, "Some useful probability distributions." Stanford University, Tech. Rep. 7050-6, 1965.

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

[38] R. Price, "A useful theorem for nonlinear devices having gaussian inputs," *IRE Transactions on Information Theory*, vol. 4, no. 2, pp. 69–72, June 1958.

[39] K. S. Narendra and K. Parthasarathy, "Identification and control of synamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 3–27, March 1990.

[40] D. P. Mandic, "A generalized normalized gradient descent algorithm," *IEEE Signal Processing Letters*, vol. 2, pp. 115–118, February 2004.

[41] W. Greblicki and M. Pawlak, "Identification of discrete hammerstein systems using kernel regression estimates," *IEEE Transactions on Automatic Control*, vol. 31, no. 1, pp. 74–77, January 1986.

[42] L. Glass and M. C. Mackey, "Pathological physiological conditions resulting from instabilities in physiological control systems," *Ann. NY. Acad. Sci*, vol. 316, pp. 214–235, 1979.