Online Dictionary Learning for Kernel LMS

Wei Gao, *Student Member, IEEE*, Jie Chen, *Member, IEEE*, Cédric Richard, *Senior Member, IEEE*, and Jianguo Huang, *Senior Member, IEEE*

***Abstract*—Adaptive filtering algorithms operating in reproducing kernel Hilbert spaces have demonstrated superiority over their linear counterpart for nonlinear system identification. Unfortunately, an undesirable characteristic of these methods is that the order of the filters grows linearly with the number of input data. This dramatically increases the computational burden and memory requirement. A variety of strategies based on dictionary learning have been proposed to overcome this severe drawback. In the literature, there is no theoretical work that strictly analyzes the problem of updating the dictionary in a time-varying environment. In this paper, we present an analytical study of the convergence behavior of the Gaussian least-mean-square algorithm 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 dictionary in an online way, by discarding the obsolete elements and adding appropriate ones. We introduce a kernel least-mean-square algorithm with ℓ_1-norm regularization to automatically perform this task. The stability in the mean of this method is analyzed, and the improvement of performance due to this dictionary adaptation is confirmed by simulations.**

***Index Terms*—Nonlinear adaptive filtering, reproducing kernel, sparsity, online forward-backward splitting.**

I. Introduction

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 Fig. 1, 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 conceptual and implementational points of view. 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., [1], [2], to series expansion methods, e.g., [3], [4]. Polynomial filters, usually called Volterra series based filters [5], and neural networks [6] have been extensively studied over the years. Volterra filters are attractive because their output is 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 [7]. 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.

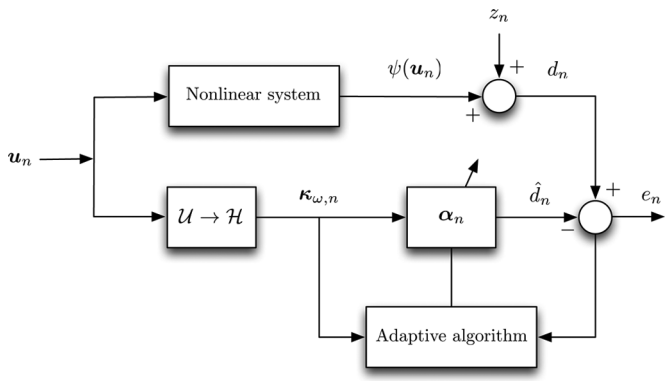

Fig. 1. Kernel-based adaptive system identification.

Recently, adaptive filtering in reproducing kernel Hilbert spaces (RKHS) has become an appealing tool in many practical fields, including biomedical engineering [8], remote sensing [9]–[12] and control [13], [14]. This framework for nonlinear system identification consists of mapping the original input 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 Fig. 1 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, and $(\mathcal{H}, \langle\cdot,\cdot\rangle_{\mathcal{H}})$ is the induced RKHS with its inner product. Usual kernels involve, e.g., the radially Gaussian and Laplacian kernels, and the q-th degree polynomial kernel. 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 $\psi^*(\,\cdot\,)$ in $\mathcal{H}$ that solves the problem

$$\psi^* = \arg\min_{\psi\in\mathcal{H}} \left\{ J(\psi) = \sum_{i=1}^{N} (d_i - \psi(\boldsymbol{u}_i))^2 + \zeta\, \Omega(\|\psi\|) \right\} \quad (1)$$

Manuscript received June 13, 2013; revised November 18, 2013, February 19, 2014; accepted March 31, 2014. Date of publication April 17, 2014; date of current version May 06, 2014. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ruixin Niu. This work was partially supported by the National Natural Science Foundation of China (61271415).

W. Gao is with the Université de Nice Sophia-Antipolis, CNRS, Observatoire de la Côte d'Azur, Nice 06103, France, and the College of Marine Engineering, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: gao_wei@mail.nwpu.edu.cn).

J. Chen and C. Richard are with the Université de Nice Sophia-Antipolis, CNRS, Observatoire de la Côte d'Azur, Nice 06103, France (e-mail: jie.chen@unice.fr; cedric.richard@unice.fr).

J. Huang is with the College of Marine Engineering, Northwestern Polytechnical University, Xi'an 710072, China (e-mail: jghuang@nwpu.edu.cn).

Digital Object Identifier 10.1109/TSP.2014.2318132

with $\Omega(\cdot)$ a real-valued monotonic regularizer on $\mathbb{R}_+$ and ζ a positive regularization constant. By virtue of the representer theorem [15], the function $\psi(\cdot)$ can be written as a kernel expansion in terms of available training data, namely, $\psi(\cdot) = \sum_{j=1}^{N} \alpha_j \, \kappa(\cdot, \boldsymbol{u}_j)$. The above optimization problem becomes

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

where $\boldsymbol{\kappa}_j$ is the $(N \times 1)$ vector with 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 (1)–(2) 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 fixed-size models of the form

$$\psi(\cdot) = \sum_{j=1}^{M} \alpha_j \, \kappa(\cdot, \boldsymbol{u}_{\omega_j}). \quad (3)$$

We call $\mathcal{D}=\{\kappa(\cdot, \boldsymbol{u}_{\omega_j})\}_{j=1}^{M}$ the dictionary, which has to be learnt from input data, and M the order of the kernel expansion by analogy with linear transversal filters. The subscript ω_j allows us to clearly distinguish dictionary elements $\boldsymbol{u}_{\omega_j}, \ldots, \boldsymbol{u}_{\omega_M}$ from 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 parameter update step. This two-step process is the essence of most adaptive filtering techniques with kernels [16].

Based on this scheme, several state-of-the-art linear methods were reconsidered to derive powerful nonlinear generalizations operating in high-dimensional RKHS [17], [18]. On the one hand, the kernel recursive least-squares algorithm (KRLS) was introduced in [19]. It can be seen as a kernel-based counterpart of the RLS algorithm, and it is characterized by a fast convergence speed at the expense of a quadratic computational complexity in M. The sliding-window KRLS and extended KRLS algorithms were successively derived in [20], [21] to improve to tracking ability of the KRLS algorithm. More recently, the KRLS tracker algorithm was introduced in [22], with ability to forget 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 [16]. On the other hand, the kernel affine projection algorithm (KAPA) and, as a particular case, the kernel normalized LMS algorithm (KNLMS), were independently introduced in [23]–[26]. The kernel least-mean-square algorithm (KLMS) was presented in [27], [28], and has attracted substantial research interest because of its linear computational complexity in M, superior tracking ability and robustness. It however converges more slowly than the KRLS algorithm. The KAPA algorithm has intermediate characteristics between the KRLS and KLMS algorithms in terms of convergence speed, computational complexity and tracking ability. A very detailed analysis of the stochastic behavior of the KLMS algorithm with Gaussian kernel was provided in [29], and a closed-form condition for convergence was recently introduced in [30]. The quantized KLMS algorithm (QKLMS) was proposed in [31], and the QKLMS algorithm with ℓ_1-norm regularization was introduced in [32]. Note that the latter uses ℓ_1-norm in order to sparsify the parameter vector $\boldsymbol{\alpha}$ in the kernel expansion (3). A subgradient approach was considered to accomplish this task, which contrasts with the more efficient forward-backward splitting algorithm recommended in [33], [34]. A recent trend within the area of adaptive filtering with kernels consists of extending all the algorithms to give them the ability to process complex input signals [35], [36]. The convergence analysis of the complex KLMS algorithm with Gaussian kernel presented in [37] is a direct application of the derivations in [29]. Finally, the quaternion kernel least-squares algorithm was recently introduced in [38].

All the above-mentioned methods 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. One of the most informative criteria uses the so-called approximate linear dependency (ALD) condition. To ensure the novelty of a candidate for becoming a new dictionary element, this criterion checks that it cannot be well approximated as a linear combination of the samples $\kappa(\cdot, \boldsymbol{u}_{\omega_j})$ that are already in the dictionary [19]. Other well-known criteria include the novelty criterion [39], the coherence criterion [24], the surprise criterion [40], and closed-ball sparsification criterion [41]. Without loss of generality, we focus on the KLMS algorithm with coherence criterion due to its simplicity and effectiveness, and because its performance are well described and understood by theoretical models [29], [30] that are exploited here. However, the dictionary update procedure studied in this paper can be adapted to the above-mentioned filtering algorithms and sparsification criteria without too much effort.

Except the above-mentioned works [32], [33], 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 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 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 [42], projection onto the ℓ_1-ball [43], or online forward-backward splitting [44]. Surprisingly, this idea was little used within the context of kernel-based adaptive filtering. To the best of our knowledge, only [33] 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. In this paper, we present an analytical study of the convergence behavior of the Gaussian least-mean-square 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 KLMS algorithm with ℓ_1-norm regularization in order to automatically perform this task. The stability of this method is analyzed and, finally, it is tested with experiments.

II. Behavior Analysis of Gaussian KLMS Algorithm With Partially Matching Dictionary

Signal reconstruction from a redundant dictionary has been extensively addressed during the last decade [45], 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 [18]. This suggests to discard obsolete dictionary elements of KLMS 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 Gaussian kernel depicted in [29] 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 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 [29]. Both models are, however, strictly equivalent in the stationary case. This simplification is one of the contributions of the paper. It might allow us to analyze other variants of the Gaussian KLMS algorithm, including the multi-kernel case, in future research works.

A. KLMS Algorithms

Several versions of the KLMS algorithm have been proposed in the literature. The two most significant versions consist of considering the problem (1) and performing gradient descent on the function $\psi(\cdot)$ in $\mathcal{H}$, or considering the problem (2) and performing gradient descent on the parameter vector $\boldsymbol{\alpha}$, respectively. The former strategy is considered in [28], [31] for instance, while the latter is applied in [24]. Both need the use of an extra mechanism for controlling the order M of the kernel expansion (3) at each time instant n. We shall now introduce such a model order selection stage, before briefly introducing the parameter update stage we recommend.

1) Dictionary Update: Coherence is a fundamental parameter that characterizes a dictionary in linear sparse approximation problems. It was redefined in [24] within the context of adaptive filtering with kernels as follows:

$$\mu = \max_{i \neq j} \left| \kappa \left(\boldsymbol{u}_{\omega_i}, \boldsymbol{u}_{\omega_j} \right) \right| \tag{4}$$

where κ is a unit-norm kernel. The coherence criterion suggests inserting the candidate input $\kappa(\cdot, \boldsymbol{u}_n)$ into the dictionary provided that its coherence remains below a given threshold μ_0

$$\max_{m=1,\ldots,M} \left| \kappa \left(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m} \right) \right| \leq \mu_0, \tag{5}$$

where μ_0 is a parameter in $[0, 1[$ determining both the level of sparsity and the coherence of the dictionary. Note that the quantization criterion introduced in [31] consists of comparing $\min_{m=1,\ldots,M} \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_m}\|_2$ with a threshold, where $\|\cdot\|_2$ denotes the ℓ_2-norm. It is thus equivalent to the original coherence criterion in the case of radial kernels such as the Gaussian one considered hereafter.[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 (5), the dictionary remains unaltered. On the other hand, if condition (5) 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) leads to the algorithm [24] recalled hereafter. For simplicity, note that we have voluntarily omitted the regularization term in (2), that is, $\zeta = 0$.

- Case 1: $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| > \mu_0$

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n} \tag{6}$$

- Case 2: $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \mu_0$

$$\boldsymbol{\alpha}_{n+1} = \begin{pmatrix} \boldsymbol{\alpha}_n \\ 0 \end{pmatrix} + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n} \tag{7}$$

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

The coherence 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 $\mathcal{U}$ ([24], proposition 2).

B. Mean Square Error Analysis

Consider the nonlinear system identification problem shown in Fig. 1, and the finite-order model (3) based on the Gaussian kernel

$$\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp\left(\frac{-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2}{2\xi^2} \right) \tag{8}$$

where ξ is the kernel bandwidth. The order M of the model (3) 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 vector. We consider that the entries of $\boldsymbol{u}_n$ can be correlated, and we denote by $\boldsymbol{R}_{\boldsymbol{uu}} = E\{\boldsymbol{u}_n \boldsymbol{u}_n{}^\top\}$ 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

$$\hat{d}_n = \boldsymbol{\alpha}_n^\top \boldsymbol{\kappa}_{\omega,n} \tag{9}$$

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

$$e_n = d_n - \hat{d}_n. \tag{10}$$

Squaring both sides of (10) and taking the expected value leads to the mean square error (MSE)

$$J_{\mathrm{ms}}(n) = E\left\{e_n^2\right\} = E\left\{d_n^2\right\} - 2\boldsymbol{p}_{\kappa d}^\top \boldsymbol{\alpha}_n + \boldsymbol{\alpha}_n^\top \boldsymbol{R}_{\kappa\kappa} \boldsymbol{\alpha}_n \tag{11}$$

[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 [46]. Decreasing of f on $\mathbb{R}_+$ ensures the equivalence between the coherence criterion and the quantization criterion.

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 [29] 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

$$\boldsymbol{\alpha}_{\rm opt} = \boldsymbol{R}_{\kappa\kappa}^{-1}\,\boldsymbol{p}_{\kappa d} \tag{12}$$

and the corresponding minimum MSE is

$$J_{\rm min} = E\left\{d_n^2\right\} - \boldsymbol{p}_{\kappa d}^{\top}\,\boldsymbol{R}_{\kappa\kappa}^{-1}\,\boldsymbol{p}_{\kappa d}. \tag{13}$$

Note that expressions (12) and (13) are the well-known Wiener solution and minimum MSE, [17], [18], respectively, where the input signal vector has been replaced by the kernelized input vector.

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

$$\|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_i}\|_2^2 + \left\|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_j}\right\|_2^2 = \boldsymbol{y}_3^{\top}\,\boldsymbol{Q}_3\,\boldsymbol{y}_3 \tag{14}$$

with

$$\boldsymbol{y}_3 = \left(\boldsymbol{u}_n{}^{\top}\;\boldsymbol{u}_{\omega_i}^{\top}\;\boldsymbol{u}_{\omega_j}^{\top}\right)^{\top} \tag{15}$$

and

$$\boldsymbol{Q}_3 = \begin{pmatrix} 2\boldsymbol{I} & -\boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} & \boldsymbol{O} \\ -\boldsymbol{I} & \boldsymbol{O} & \boldsymbol{I} \end{pmatrix} \tag{16}$$

where $\boldsymbol{I}$ is the $(q \times q)$ identity matrix, and $\boldsymbol{O}$ is the $(q \times q)$ null matrix. From ([47], 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

$$\psi_z(s) = E\{e^{sz}\} = \det\{\boldsymbol{I} - 2s\boldsymbol{Q}\boldsymbol{R}_y\}^{-1/2}. \tag{17}$$

Making $s = -1/(2\xi^2)$ in (17), we find that the (i,j)-th element of $\boldsymbol{R}_{\kappa\kappa}$ is given by

$$[\boldsymbol{R}_{\kappa\kappa}]_{ij} = \begin{cases} r_{\rm md} = \det\{\boldsymbol{I}_3 + \boldsymbol{Q}_3\,\boldsymbol{R}_3(i,j)/\xi^2\}^{-1/2}, & i = j \\ r_{\rm od} = \det\{\boldsymbol{I}_3 + \boldsymbol{Q}_3\,\boldsymbol{R}_3(i,j)/\xi^2\}^{-1/2}, & i \neq j \end{cases} \tag{18}$$

with $1 \leq i,\, j \leq M$, and $r_{\rm md}$ and $r_{\rm od}$ are the main-diagonal and off-diagonal entries of $\boldsymbol{R}_{\kappa\kappa}$, respectively. In (18), $\boldsymbol{R}_p$ is the $(pq \times pq)$ correlation matrix of vector $\boldsymbol{y}_p$ (see expression (19) 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

$$\boldsymbol{R}_3(i,j) = \begin{pmatrix} \boldsymbol{R}_{\boldsymbol{uu}} & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,i) & \boldsymbol{R}_{\mathcal{D}}(i,j) \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,j) & \boldsymbol{R}_{\mathcal{D}}(j,j) \end{pmatrix} \tag{19}$$

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 [29], the formulations (18), (19), 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. To the best of our knowledge, only [32], [33] 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 [33] was proposed in the multi-kernel context. On the other hand, the algorithm [32] 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 (19) is given by

$$\boldsymbol{R}_{\mathcal{D}}(i,j) = \begin{cases} \boldsymbol{R}_{\boldsymbol{uu}}, & 1 \leq i = j \leq L \\ \tilde{\boldsymbol{R}}_{\boldsymbol{uu}}, & L < i = j \leq M \\ \boldsymbol{O}, & 1 \leq i \neq j \leq M \end{cases} \tag{20}$$

which allows to calculate the correlation matrix $\boldsymbol{R}_{\kappa\kappa}$ of the kernelized input via (18). Note that $\boldsymbol{R}_{\mathcal{D}}(i,j)$ in (19) reduces to $\delta_{ij}\,\boldsymbol{R}_{\boldsymbol{uu}}$, with $\delta_{ij} = 1$ if $(i = j)$, otherwise 0, in the case $(L = M)$ considered in [29].

C. 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 $J_{\rm ms}(n)$ defined in (11).

1) Mean Weight Behavior: The weight update equation of KLMS algorithm is given by

$$\boldsymbol{\alpha}_{n+1} = \boldsymbol{\alpha}_n + \eta\,e_n\,\boldsymbol{\kappa}_{\omega,n} \tag{21}$$

where η is the step size. Defining the weight error vector $\boldsymbol{v}_n = \boldsymbol{\alpha}_n - \boldsymbol{\alpha}_{\rm opt}$ leads to the weight error vector update equation

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \eta\,e_n\,\boldsymbol{\kappa}_{\omega,n}. \tag{22}$$

From (9) and (10), and the definition of $\boldsymbol{v}_n$, the error equation is given by

$$e_n = d_n - \boldsymbol{\kappa}_{\omega,n}^{\top}\,\boldsymbol{v}_n - \boldsymbol{\kappa}_{\omega,n}^{\top}\,\boldsymbol{\alpha}_{\rm opt} \tag{23}$$

and the optimal estimation error is

$$e_n^o = d_n - \boldsymbol{\kappa}_{\omega,n}^{\top}\,\boldsymbol{\alpha}_{\rm opt}. \tag{24}$$

Substituting (23) into (22) yields

$$\boldsymbol{v}_{n+1} = \boldsymbol{v}_n + \eta\,d_n\,\boldsymbol{\kappa}_{\omega,n} - \eta\,\boldsymbol{\kappa}_{\omega,n}^{\top}\,\boldsymbol{v}_n\,\boldsymbol{\kappa}_{\omega,n} - \eta\,\boldsymbol{\kappa}_{\omega,n}^{\top}\,\boldsymbol{\alpha}_{\rm opt}\,\boldsymbol{\kappa}_{\omega,n}. \tag{25}$$

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 [48], and shown to be less restrictive than the independence assumption [17]. 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 (25) and using these two assumptions yields

$$E\{\boldsymbol{v}_{n+1}\} = (\boldsymbol{I} - \eta\, \boldsymbol{R}_{\kappa\kappa})\, E\{\boldsymbol{v}_n\}. \tag{26}$$

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

2) Mean Square Error Behavior: Using (23) and the MIA, the second-order moments of the weights are related to the MSE through [17]

$$J_{\rm ms}(n) = J_{\min} + \mathrm{trace}\{\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_v(n)\} \tag{27}$$

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 $\mathrm{trace}\{\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_v(n)\}$ is the excess MSE (EMSE). The analysis of the MSE behavior (27) 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 [29]. Using simplifying assumptions derived from the MIA, it reduces to the following recursion

$$\begin{aligned}\boldsymbol{C}_v(n+1) \approx \boldsymbol{C}_v(n) - \eta\,(\boldsymbol{R}_{\kappa\kappa}\boldsymbol{C}_v(n) + \boldsymbol{C}_v(n)\boldsymbol{R}_{\kappa\kappa}) \\ +\eta^2\,\boldsymbol{T}(n) + \eta^2\,\boldsymbol{R}_{\kappa\kappa}J_{\min}\end{aligned} \tag{28a}$$

with

$$\boldsymbol{T}(n) = E\left\{\boldsymbol{\kappa}_{\omega,n}\,\boldsymbol{\kappa}_{\omega,n}^{\top}\,\boldsymbol{v}_n\,\boldsymbol{v}_n^{\top}\,\boldsymbol{\kappa}_{\omega,n}\,\boldsymbol{\kappa}_{\omega,n}^{\top}\right\}. \tag{28b}$$

The evaluation of expectation (28b) is an important step in the analysis. It leads to extensive calculus if proceeding as in [29] 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 paper, 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 (28b) yields

$$\begin{aligned}[\boldsymbol{T}(n)]_{ij} \approx \sum_{\ell=1}^{M}\sum_{p=1}^{M} E\{\kappa_{\omega,n}(i)\,\kappa_{\omega,n}(j)\,\kappa_{\omega,n}(\ell)\,\kappa_{\omega,n}(p)\} \\ \times[\boldsymbol{C}_v(n)]_{\ell p}.\end{aligned} \tag{29}$$

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

$$[\boldsymbol{T}(n)]_{ij} \approx \mathrm{trace}\{\boldsymbol{K}(i,j)\,\boldsymbol{C}_v(n)\} \tag{30}$$

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

$$\begin{aligned}z = \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_i}\|_2^2 + \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_j}\|_2^2 + \|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_\ell}\|_2^2 \\ +\|\boldsymbol{u}_n - \boldsymbol{u}_{\omega_p}\|_2^2.\end{aligned} \tag{31}$$

Using expression (17) leads us to

$$[\boldsymbol{K}(i,j)]_{\ell,p} = [\det\{\boldsymbol{I}_5 + \boldsymbol{Q}_5\,\boldsymbol{R}_5(i,j,\ell,p)/\xi^2\}]^{-1/2} \tag{32}$$

with

$$\boldsymbol{Q}_5 = \begin{pmatrix} 4\boldsymbol{I} & -\boldsymbol{I} & -\boldsymbol{I} & -\boldsymbol{I} & -\boldsymbol{I} \\ -\boldsymbol{I} & \boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} \\ -\boldsymbol{I} & \boldsymbol{O} & \boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} \\ -\boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{I} & \boldsymbol{O} \\ -\boldsymbol{I} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{I} \end{pmatrix}, \tag{33}$$

and

$$\begin{aligned}&\boldsymbol{R}_5(i,j,\ell,p) \\ &= \begin{pmatrix} \boldsymbol{R}_{\boldsymbol{uu}} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,i) & \boldsymbol{R}_{\mathcal{D}}(i,j) & \boldsymbol{R}_{\mathcal{D}}(i,\ell) & \boldsymbol{R}_{\mathcal{D}}(i,p) \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,j) & \boldsymbol{R}_{\mathcal{D}}(j,j) & \boldsymbol{R}_{\mathcal{D}}(j,\ell) & \boldsymbol{R}_{\mathcal{D}}(j,p) \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,\ell) & \boldsymbol{R}_{\mathcal{D}}(j,\ell) & \boldsymbol{R}_{\mathcal{D}}(\ell,\ell) & \boldsymbol{R}_{\mathcal{D}}(\ell,p) \\ \boldsymbol{O} & \boldsymbol{R}_{\mathcal{D}}(i,p) & \boldsymbol{R}_{\mathcal{D}}(j,p) & \boldsymbol{R}_{\mathcal{D}}(\ell,p) & \boldsymbol{R}_{\mathcal{D}}(p,p) \end{pmatrix},\end{aligned} \tag{34}$$

which uses the same block definition as in (20). 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 [29]. This expression concludes the calculation.

D. Steady-State Behavior

We shall now determine the steady-state of the recursion (28a). 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 $\mathrm{vec}\{\cdot\}$ 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 $\boldsymbol{c}_v(n) = \mathrm{vec}\{\boldsymbol{C}_v(n)\}$ and $\boldsymbol{r}_{\kappa\kappa} = \mathrm{vec}\{\boldsymbol{R}_{\kappa\kappa}\}$, it can be verified that (28a) can be rewritten as

$$\boldsymbol{c}_v(n+1) = \boldsymbol{G}\,\boldsymbol{c}_v(n) + \eta^2 J_{\min}\,\boldsymbol{r}_{\kappa\kappa} \tag{35}$$

with

$$\boldsymbol{G} = \boldsymbol{I} - \eta(\boldsymbol{G}_1 + \boldsymbol{G}_2) + \eta^2\boldsymbol{G}_3. \tag{36}$$

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 $\boldsymbol{C}_v(n)\boldsymbol{R}_{\kappa\kappa}$. It is a block-diagonal matrix, with $\boldsymbol{R}_{\kappa\kappa}$ on its diagonal. It can thus be written as $\boldsymbol{G}_1 = \boldsymbol{I} \otimes \boldsymbol{R}_{\kappa\kappa}$, 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 $\boldsymbol{R}_{\kappa\kappa} \otimes \boldsymbol{I}$;
- $\boldsymbol{G}_3 = \mathrm{vec}\{\boldsymbol{T}(n)\}$ with $\boldsymbol{T}(n)$ defined as in (30), namely,

$$[\boldsymbol{G}_3]_{i+(j-1)M,\ell+(p-1)M} = [\boldsymbol{K}(i,j)]_{\ell,p} \tag{37}$$

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

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 (35) is given by

$$\boldsymbol{c}_v(n) = \boldsymbol{G}^n\,[\boldsymbol{c}_v(0) - \boldsymbol{c}_v(\infty)] + \boldsymbol{c}_v(\infty) \tag{38}$$

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

$$\boldsymbol{c}_v(\infty) = \eta^2\,J_{\min}\,(\boldsymbol{I} - \boldsymbol{G})^{-1}\,\boldsymbol{r}_{\kappa\kappa}. \tag{39}$$

From (27), the steady-state MSE is finally given by

$$J_{\rm ms}(\infty) = J_{\min} + \mathrm{trace}\{\boldsymbol{R}_{\kappa\kappa}\,\boldsymbol{C}_v(\infty)\} \tag{40}$$

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

TABLE I
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
				$\{10@0.15\} \cup \{10@0.35\}$	-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
				$\{10@0.15\} \cup \{10@0.35\}$	-11.01	-10.99	-35.81	31614

TABLE II
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	$\sqrt{0.0656} \rightarrow \sqrt{0.0156}$	$\{15@\sqrt{0.0656}\}$	$\{15@\sqrt{0.0656}\}$	-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	$\sqrt{0.0156} \rightarrow \sqrt{0.0656}$	$\{15@\sqrt{0.0156}\}$	$\{15@\sqrt{0.0156}\}$	-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

III. SIMULATION RESULTS

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 I and II, 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'.

Example 1: Consider the problem studied in [29], [49], [50], for which

$$\begin{cases} y(n) = \dfrac{y(n-1)}{1+y^2(n-1)} + u^3(n-1) \\ d(n) = y(n) + z(n) \end{cases} \tag{41}$$

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 I 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

$$\|\boldsymbol{c}_v(\infty) - \boldsymbol{c}_v(n_\epsilon)\|_2 \leq 10^{-3}. \tag{42}$$

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 Figs. 2 and 3.

Example 2: Consider the nonlinear dynamic system studied in [29], [51] where the input signal was a sequence of statistically independent vectors

$$\boldsymbol{u}_n = [u_1(n)\ u_2(n)]^\top \tag{43}$$

with correlated samples satisfying $u_1(n) = 0.5u_2(n) + v_u(n)$. 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

$$y(n) = \boldsymbol{a}^\top \boldsymbol{u}_n - 0.2\, y(n-1) + 0.35\, y(n-2) \tag{44}$$

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

$$\varphi(y(n)) = \begin{cases} \dfrac{y(n)}{3[0.1 + 0.9\, y^2(n)]^{1/2}} & \text{for } y(n) \geq 0 \\ \dfrac{-y^2(n)[1 - \exp(0.7y(n))]}{3} & \text{for } y(n) < 0 \end{cases} \tag{45}$$

$$d(n) = \varphi(y(n)) + z(n) \tag{46}$$

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

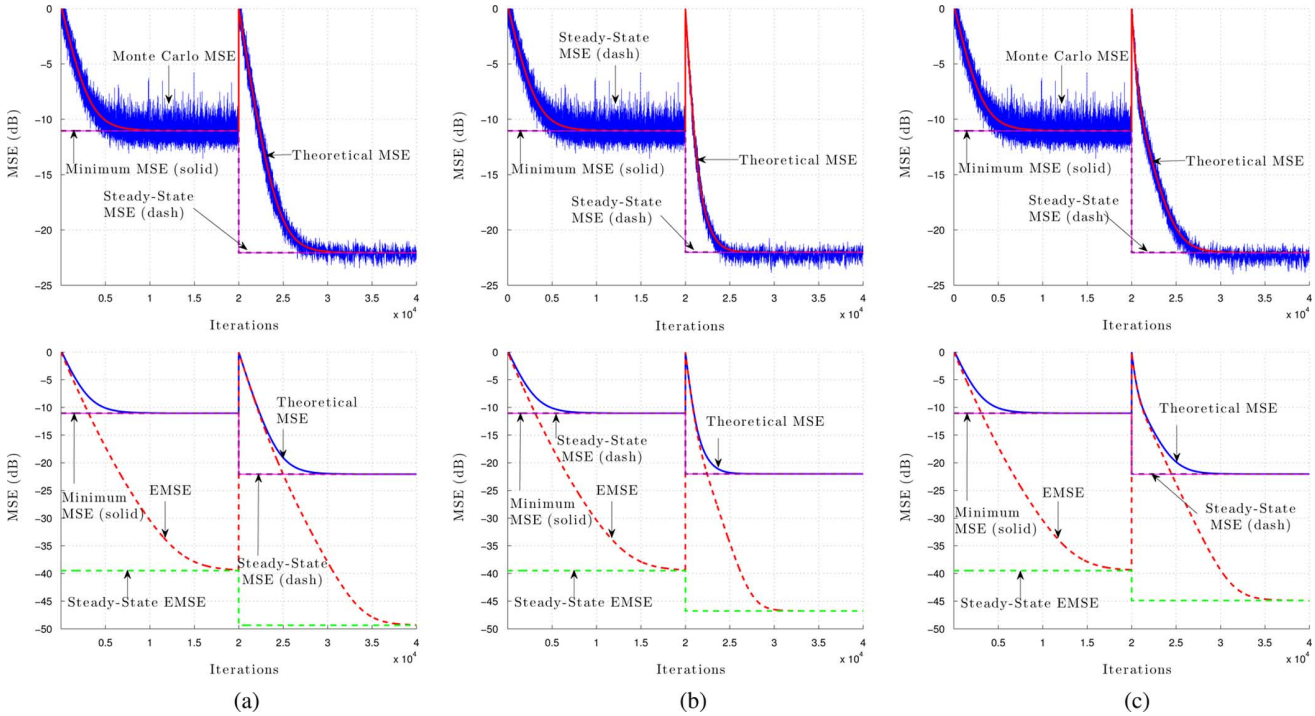

Fig. 2. Learning curves for Example 1 where $\sigma_u : 0.35 \rightarrow 0.15$ and $\mathcal{D}_1 = \{10@0.35\}$. See the first row of Table I. (a) $\mathcal{D}_2 = \{10@0.35\}$. (b) $\mathcal{D}_2 = \{10@0.15\}$. (c) $\mathcal{D}_2 = \{10@0.15\} \cup \{10@0.35\}$.

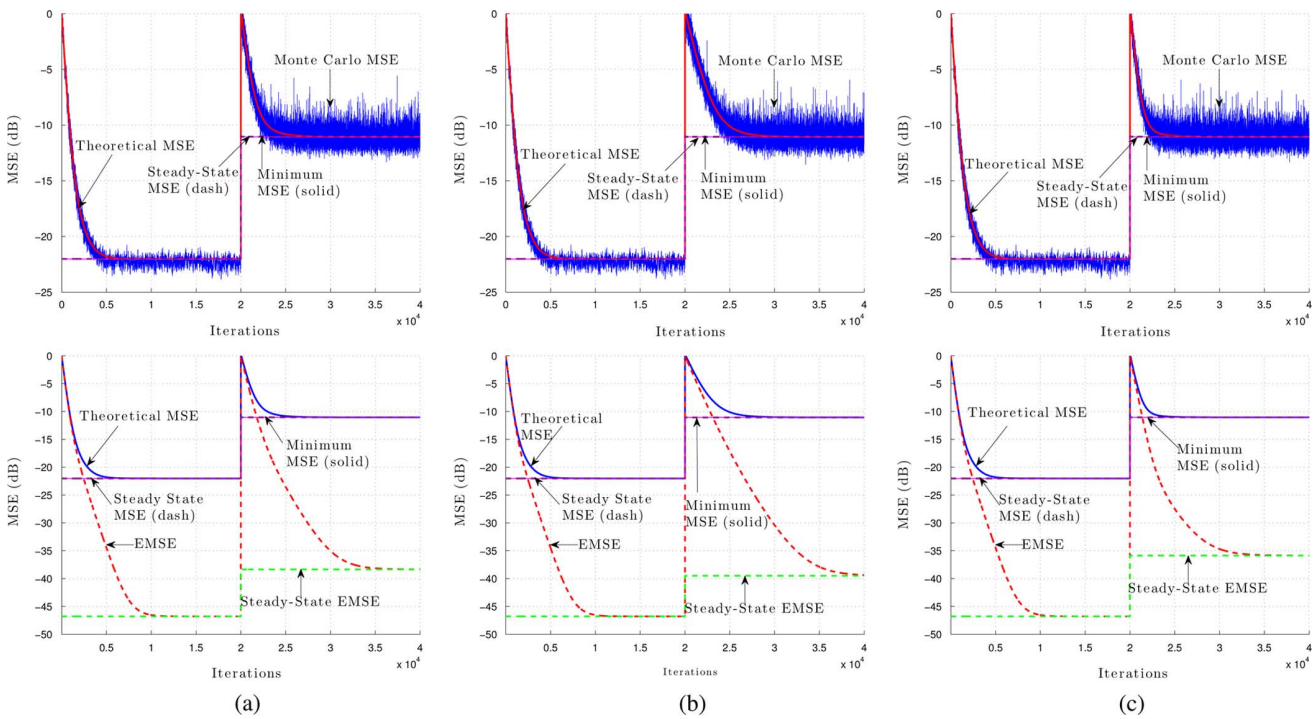

Fig. 3. Learning curves for Example 1 where $\sigma_u : 0.15 \rightarrow 0.35$ and $\mathcal{D}_1 = \{10@0.15\}$. See the second row of Table I. (a) $\mathcal{D}_2 = \{10@0.15\}$. (b) $\mathcal{D}_2 = \{10@0.35\}$. (c) $\mathcal{D}_2 = \{10@0.15\} \cup \{10@0.35\}$.

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 II 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 Figs. 4 and 5.

1) 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

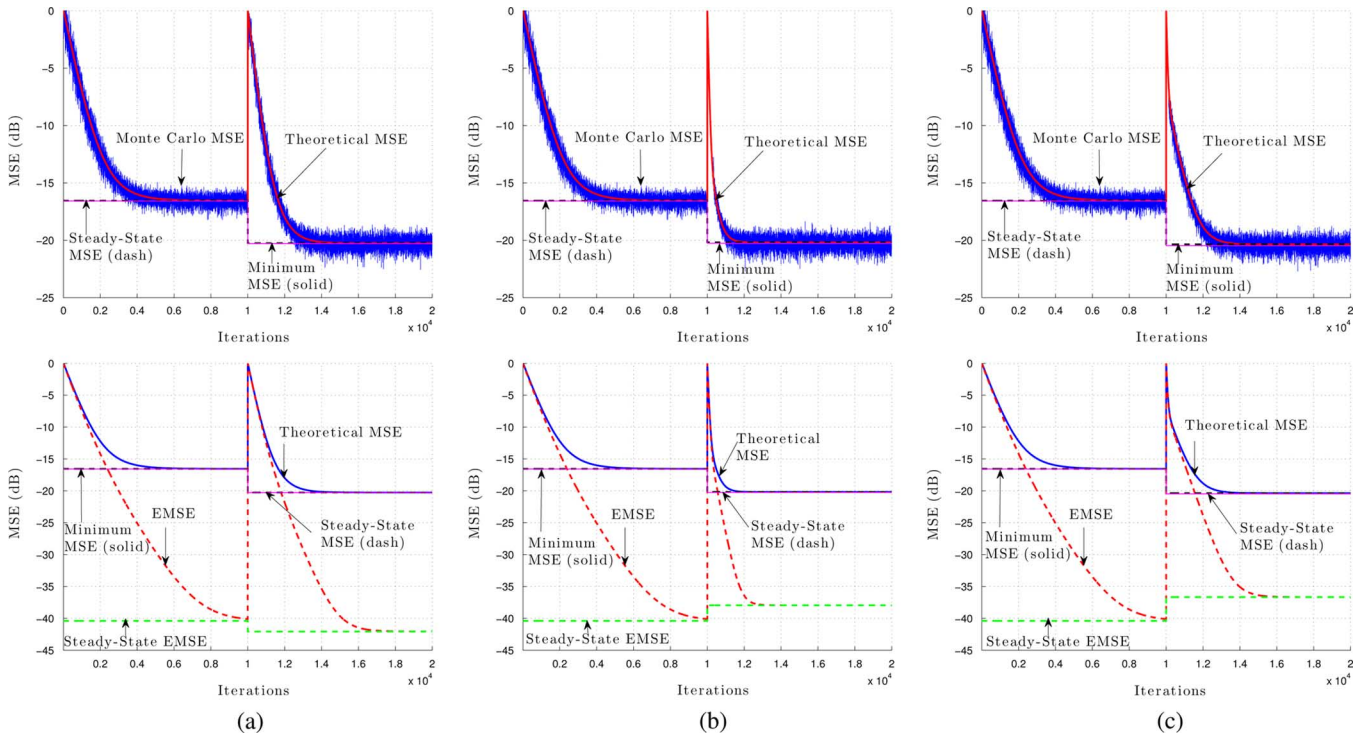

Fig. 4. Learning curves for Example 2 with $\sigma_{v_2}, \sigma_{v_u} : \sqrt{0.0656} \rightarrow \sqrt{0.0156}$ and $\mathcal{D}_1 = \{15@\sqrt{0.0656}\}$. See the first row of Table II. (a) $\mathcal{D}_2 = \{15@\sqrt{0.0656}\}$. (b) $\mathcal{D}_2 = \{15@\sqrt{0.0156}\}$. (c) $\mathcal{D}_2 = \{15@\sqrt{0.0156}\} \cup \{15@\sqrt{0.0656}\}$.

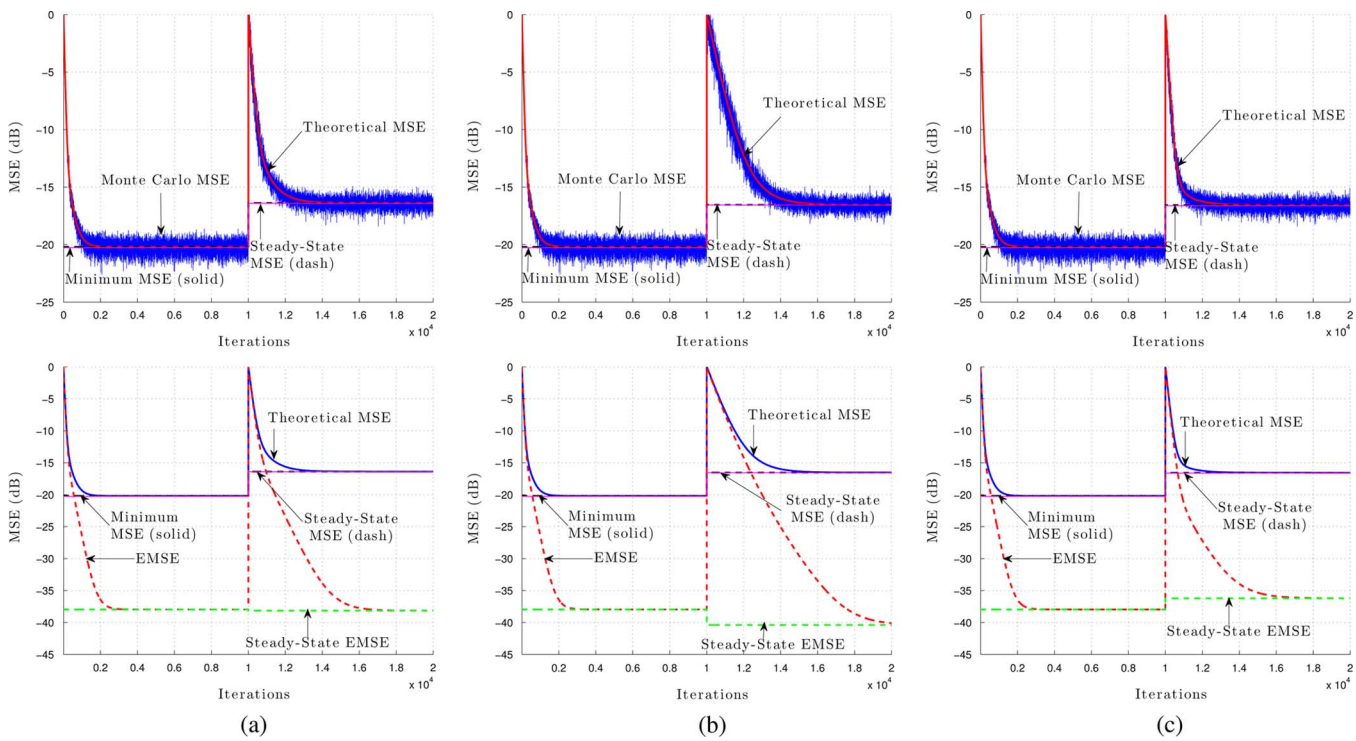

Fig. 5. Learning curves for Example 2 with $\sigma_{v_2}, \sigma_{v_u} : \sqrt{0.0156} \rightarrow \sqrt{0.0656}$ and $\mathcal{D}_1 = \{15@\sqrt{0.0156}\}$. See the second row of Table II. (a) $\mathcal{D}_2 = \{15@\sqrt{0.0156}\}$. (b) $\mathcal{D}_2 = \{15@\sqrt{0.0656}\}$. (c) $\mathcal{D}_2 = \{15@\sqrt{0.0156}\} \cup \{15@\sqrt{0.0656}\}$.

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.

IV. 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 elements and adding appropriate ones.

A. Forward-Backward Splitting Method in a Nutshell

Consider first the following optimization problem [52]

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \{Q(\boldsymbol{\alpha}) = J(\boldsymbol{\alpha}) + \lambda\Omega(\boldsymbol{\alpha})\} \tag{47}$$

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 [53]. 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,

$$Q_\eta(\boldsymbol{\alpha},\boldsymbol{\alpha}_n) = J(\boldsymbol{\alpha}_n) + \nabla J(\boldsymbol{\alpha}_n)^\top(\boldsymbol{\alpha}-\boldsymbol{\alpha}_n) + \frac{1}{2\eta}\|\boldsymbol{\alpha}-\boldsymbol{\alpha}_n\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}) \tag{48}$$

since $Q(\boldsymbol{\alpha}) \le Q_\eta(\boldsymbol{\alpha},\boldsymbol{\alpha}_n)$ for any $\eta \le \eta_0$. 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

$$\boldsymbol{\alpha}_{n+1} = \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \left\{ \lambda\Omega(\boldsymbol{\alpha}) + \frac{1}{2\eta}\|\boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}}_n\|_2^2 \right\} \tag{49}$$

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 (49) 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 (49) to a global minimum is ensured if $1/\eta$ is a Lipschitz constant of the gradient $\nabla J(\boldsymbol{\alpha})$. See [52]. In the case $J(\boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{d}-\boldsymbol{K}\boldsymbol{\alpha}\|_2^2$ considered in (2), 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 (47) is to require that [53]

$$0 < \eta < 2/\mathrm{eig}_{\max}\{\boldsymbol{K}^\top\boldsymbol{K}\} \tag{50}$$

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 [54]. 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).

B. Application to KLMS Algorithm

In order to automatically discard the irrelevant elements from the dictionary $\mathcal{D}$, let us consider the minimization problem (2) with the sparsity-promoting convex regularization function $\Omega(\cdot)$

$$\boldsymbol{\alpha}^* = \arg\min_{\boldsymbol{\alpha}\in\mathbb{R}^N} \left\{ Q(\boldsymbol{\alpha}) = \|\boldsymbol{d}-\boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \lambda\Omega(\boldsymbol{\alpha}) \right\} \tag{51}$$

where $\boldsymbol{K}$ is the $(N\times N)$ Gram matrix with (i,j)-th entry $\kappa(\boldsymbol{u}_i,\boldsymbol{u}_j)$. Problem (51) is of the form (47), 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 [52]

$$(\mathrm{Prox}_{\lambda\eta\|\cdot\|_1}(\boldsymbol{\alpha}))(m) = \mathrm{sign}\{\alpha(m)\}\max\{|\alpha(m)|-\lambda\eta, 0\}. \tag{52}$$

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

$$\big(\mathrm{Prox}_{\lambda\eta\Omega_a(\cdot)}(\boldsymbol{\alpha})\big)(m) = \mathrm{sign}\{\alpha(m)\}\max\{|\alpha(m)|-\lambda\eta\, w_m, 0\}. \tag{53}$$

This regularization function has been proven to be more consistent than the usual ℓ_1-norm [55], 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), and ϵ_α a small constant to prevent the denominator from vanishing [56]. 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 [42] 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 [33] in the multi-kernel adaptive filtering context. The stability of

Algorithm 1: FOBOS-KLMS

1: **Initialization**
Select the step size η, and the parameters of the kernel;
Insert $\kappa(\cdot, \boldsymbol{u}_1)$ into the dictionary, $\boldsymbol{\alpha}_1 = 0$.
2: **for** $n = 1, 2, \ldots,$ do
3: **if** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| > \mu_0$
Compute $\boldsymbol{\kappa}_{\omega,n}$ and $\hat{\boldsymbol{\alpha}}_n$ using (6);
4: **elseif** $\max_{m=1,\ldots,M} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_m})| \leq \mu_0$
Incorporate $\kappa(\cdot, \boldsymbol{u}_n)$ into the dictionary;
Compute $\boldsymbol{\kappa}_{\omega,n}$ and $\hat{\boldsymbol{\alpha}}_n$ using (7);
5: **end if**
6: $\boldsymbol{\alpha}_n = \mathrm{Prox}_{\lambda\eta\Omega(\,\cdot\,)}(\hat{\boldsymbol{\alpha}}_n)$ using (52) or (53);
7: Remove $\kappa(\cdot, \boldsymbol{u}_{\omega_m})$ from the dictionary if $\alpha_n(m) = 0$.
8: The solution is given as

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

9: **end for**

this method is analyzed in the next subsection, which is an additional contribution of this paper.

C. Stability in the Mean

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

$$\boldsymbol{\alpha}_n = \boldsymbol{\alpha}_{n-1} + \eta\, e_n\, \boldsymbol{\kappa}_{\omega,n} - \boldsymbol{f}_{n-1} \tag{54}$$

with

$$f_{n-1}(m) = \begin{cases} \lambda\eta\, \mathrm{sign}(\hat{\boldsymbol{\alpha}}_{n-1}(m)) & \text{if} \quad |\hat{\boldsymbol{\alpha}}_{n-1}(m)| \geq \lambda\eta \\ \hat{\boldsymbol{\alpha}}_{n-1}(m) & \text{otherwise} \end{cases} \tag{55}$$

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

$$\mathrm{sign}(\alpha) = \begin{cases} \alpha/|\alpha| & \alpha \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{56}$$

Up to a variable change in λ, the general form (54), (55) remains the same with the regularization function (53). Note that the sequence $|f_{n-1}(m)|$ is bounded, by $\lambda\eta$ for the operator (52), and by $\lambda\eta/\epsilon_\alpha$ for the operator (53).

Theorem 1: Assume MIA holds. For any initial condition $\boldsymbol{\alpha}_0$, the KLMS algorithm with sparsity promoting regularization (52) and (53) asymptotically converges in the mean sense if the step-size η is chosen to satisfy

$$0 < \eta < 2/\mathrm{eig}_{\max}\{\boldsymbol{R}_{\kappa\kappa}\} \tag{57}$$

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 $\boldsymbol{R}_{\kappa\kappa}$.

To prove this theorem, we observe that the recursion (22) for the weight error vector $\boldsymbol{v}_n$ becomes

$$\boldsymbol{v}_n = \boldsymbol{v}_{n-1} - \eta\, \boldsymbol{\kappa}_{\omega,n}\, (\boldsymbol{\kappa}_{\omega,n}\, \boldsymbol{v}_{n-1} + e_n^o) - \boldsymbol{f}_{n-1}. \tag{58}$$

Taking the expected value of both sides, and using the same assumptions as for (26), leads to

$$E\{\boldsymbol{v}_n\} = (\boldsymbol{I} - \eta\, \boldsymbol{R}_{\kappa\kappa})^n E\{\boldsymbol{v}_0\} + \sum_{i=0}^{n-1} (\boldsymbol{I} - \eta\, \boldsymbol{R}_{\kappa\kappa})^i E\{\boldsymbol{f}_{n-i-1}\} \tag{59}$$

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 (57) since

$$\nu = \max_{m=1,\ldots,M} |1 - \eta\, \mathrm{eig}_m\{\boldsymbol{R}_{\kappa\kappa}\}| \tag{60}$$

where $\mathrm{eig}_m\{\boldsymbol{R}_{\kappa\kappa}\}$ is the m-th eigenvalue of $\boldsymbol{R}_{\kappa\kappa}$. Let us show now that condition (57) also implies that the second term on the r.h.s. of (59) 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

$$\begin{aligned} &\|(\boldsymbol{I} - \eta\, \boldsymbol{R}_{\kappa\kappa})^i E\{\boldsymbol{f}_{n-i-1}\}\|_2 \\ &\quad \leq \|(\boldsymbol{I} - \eta\, \boldsymbol{R}_{\kappa\kappa})^i\|_2 E\{\|\boldsymbol{f}_{n-i-1}\|_2\} \\ &\quad \leq \sqrt{M}\, \nu^i\, f_{\max} \end{aligned} \tag{61}$$

where $f_{\max} = \lambda\eta$ or $\lambda\eta/\epsilon_\alpha$, depending if one uses the regularization function (52) or (53). Condition (57) implies that $\nu < 1$ and, as a consequence,

$$\sum_{i=0}^{n-1} \|(\boldsymbol{I} - \eta\, \boldsymbol{R}_{\kappa\kappa})^i E\{\boldsymbol{f}_{n-i-1}\}\|_2 \leq \frac{\sqrt{M}\, f_{\max}}{1 - \nu}. \tag{62}$$

The second term on the r.h.s. of (59) is an absolutely convergent series. This implies that it is a convergent series. Because the two terms of (59) are convergent series, we finally conclude that $E\{\boldsymbol{v}_n\}$ converges to a steady-state value if condition (57) is satisfied. Before concluding this section, it should be noticed that we have shown in [29] that

$$\mathrm{eig}_{\max}\{\boldsymbol{R}_{\kappa\kappa}\} = r_{\mathrm{md}} + (M - 1)\, r_{\mathrm{od}}. \tag{63}$$

Parameters r_{md} and r_{od} are given by expression (18) in the case of a possibly partially matching dictionary.

D. Simulation Results of Proposed Algorithm

We shall now illustrate the good performance of the FOBOS-KLMS algorithm with the two examples considered in Section II. Experimental settings were unchanged, and the results were averaged over 200 Monte Carlo runs. The coherence threshold μ_0 in Algorithm 1 was set to 0.01.

One can observe in Figs. 7 and 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 Figs. 6 and 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. Figs. 6 to 9 clearly illustrate the merits of the FOBOS-KLMS algorithm with the

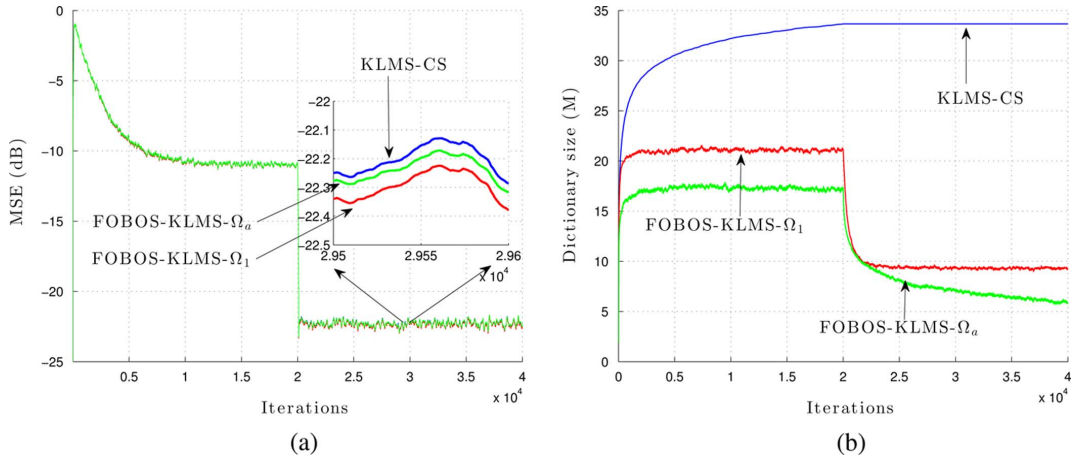

Fig. 6. Learning curves for Example 1 where $\sigma_u : 0.35 \rightarrow 0.15$. (a) MSE. (b) Evolution of the size of dictionary.

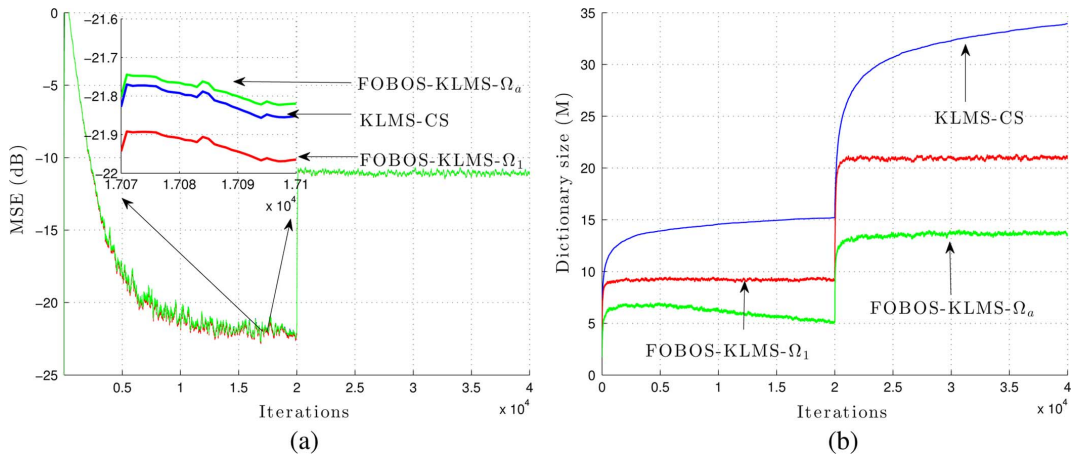

Fig. 7. Learning curves for Example 1 where $\sigma_u : 0.15 \rightarrow 0.35$. (a) MSE. (b) Evolution of the size of dictionary.

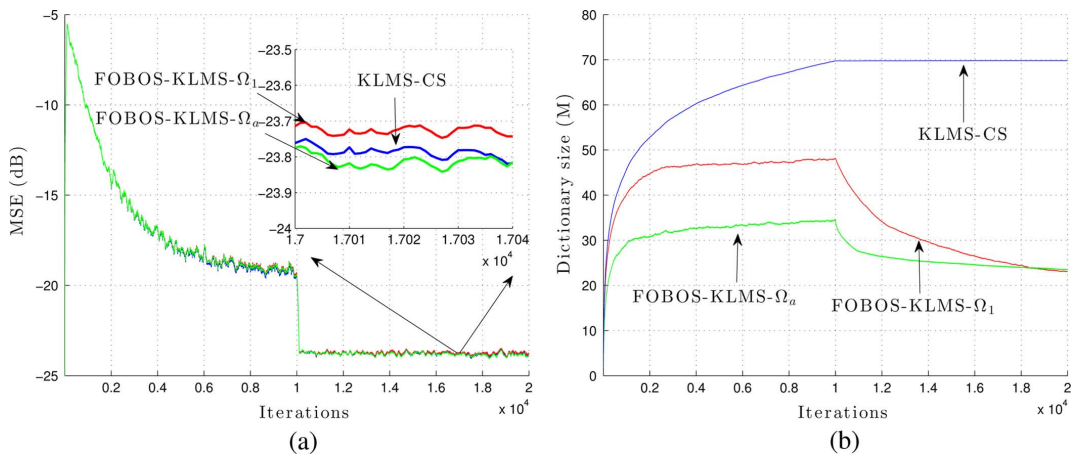

Fig. 8. Learning curves for Example 2 with $\sigma_{u_2}, \sigma_{v_u} : \sqrt{0.0656} \rightarrow \sqrt{0.0156}$. (a) MSE. (b) Evolution of the size of dictionary.

regularizations (52) and (53). 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 its performance.

V. Conclusion

In this paper, we presented an analytical study of the convergence behavior of the Gaussian least-mean-square algorithm in the case where the statistics of the dictionary elements only partially match the statistics of the input data. This allowed us to emphasize the need for updating the dictionary in an online way, by discarding the obsolete elements and adding appropriate ones. We introduced 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 characteristics of the input signal. The

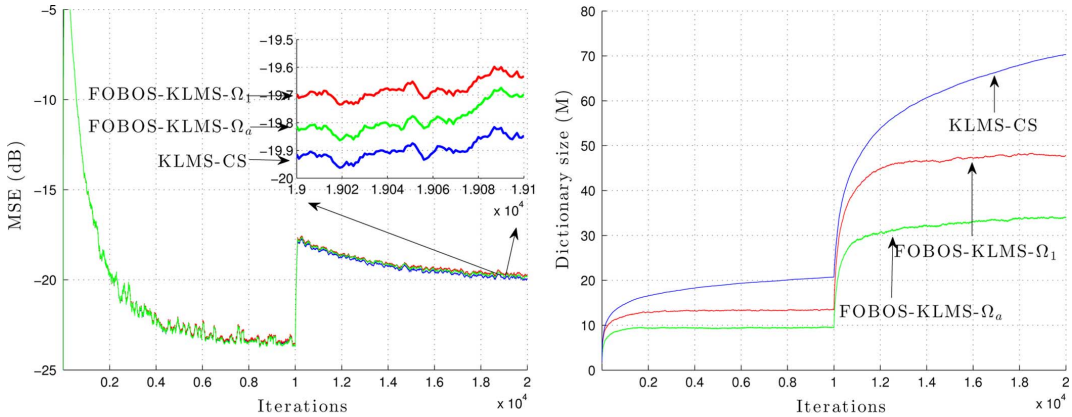

Fig. 9. Learning curves for Example 2 with $\sigma_{u_2}, \sigma_{v_u}$: $\sqrt{0.0156} \rightarrow \sqrt{0.0656}$. (a) MSE. (b) Evolution of the size of dictionary.

stability in the mean of this method was analyzed, and a condition on the step-size for convergence was derived. The merits of FOBOS-KLMS were illustrated by simulation examples.

REFERENCES

[1] S. W. Nam and E. J. Powers, "Application of higher order spectral analysis to cubically nonlinear system identification," *IEEE Signal Process. Mag.*, vol. 42, no. 7, pp. 2124–2135, 1994.

[2] C. L. Nikias and A. P. Petropulu, *Higher-Order Spectra Analysis—A Nonlinear Signal Processing Framework*. Englewood Cliffs, NJ: Prentice-Hall, 1993.

[3] M. Schetzen, *The Volterra and Wiener Theory of the Nonlinear Systems*. New York, NY, USA: Wiley, 1980.

[4] N. Wiener, *Nonlinear Problems in Random Theory*. New York, NY, USA: Wiley, 1958.

[5] V. J. Mathews and G. L. Sicuranze, *Polynomial Signal Processing*. New York, NY, USA: Wiley, 2000.

[6] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1999.

[7] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superpositions of continuous functions of one variable and addition," *Dokl. Akad. Nauk USSR*, vol. 114, pp. 953–956, 1957.

[8] *Kernel Methods in Bioengineering, Signal and Image Processing*, G. Camps-Valls, J. L. Rojo-Alvarez, and M. Martinez-Ramon, Eds. Hershey, PA, USA: IGI Global, 2007.

[9] , G. Camps-Valls and L. Bruzzone, Eds., *Kernel Methods for Remote Sensing Data Analysis*. New York, NY, USA: Wiley, 2009.

[10] J. Chen, C. Richard, and P. Honeine, "Nonlinear unmixing of hyperspectral data based on a linear-mixture/nonlinear-fluctuation model," *IEEE Trans. Signal Process.*, vol. 61, no. 2, pp. 480–492, 2013.

[11] J. Chen, C. Richard, and P. Honeine, "Nonlinear estimation of material abundances in hyperspectral images with ℓ_1-norm spatial regularization," *IEEE Trans. Geosci. Remote Sens.*, vol. 52, no. 5, pp. 2654–2655, 2013.

[12] 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 Process. Mag.*, vol. 31, no. 1, pp. 82–94, 2014.

[13] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 18, no. 4, pp. 973–992, 2007.

[14] D. Nguyen-Tuong and J. Peters, "Online Kernel-based learning for task-space tracking robot control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 9, pp. 1417–1425, 2012.

[15] B. Schölkopf, R. Herbrich, and R. Williamson, "A generalized representer theorem," NeuroCOLT, Royal Holloway College, Univ. of London, London, U.K., Tech. Rep. NC2-TR-2000-81, 2000.

[16] W. Liu, J. C. Príncipe, and S. Haykin, *Kernel Adaptive Filtering*. Hoboken, NJ, USA: Wiley, 2010.

[17] A. H. Sayed, *Fundamentals of Adaptive Filtering*. New York, NY, USA: Wiley, 2003.

[18] S. Haykin, *Adaptive Filter Theory*, 2nd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.

[19] Y. Engel, S. Mannor, and R. Meir, "The Kernel recursive least squares," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, 2004.

[20] S. V. Vaerenbergh, J. Vía, and I. Santamaría, "A sliding-window Kernel RLS algorithm and its application to nonlinear channel identification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2006, pp. 789–792.

[21] W. Liu, I. M. Park, Y. Wang, and J. Príncipe, "Extended Kernel recursive least squares algorithm," *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 3801–3814, 2009.

[22] S. V. Vaerenbergh, M. Lázaro-Gredilla, and I. Santamaría, "Kernel recursive least-squares tracker for time-varying regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1313–1326, 2012.

[23] P. Honeine, C. Richard, and J.-C. M. Bermudez, "On-line nonlinear sparse approximation of functions," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, 2007, pp. 956–960.

[24] C. Richard, J.-C. M. Bermudez, and P. Honeine, "Online prediction of time series data with Kernels," *IEEE Trans. Signal Process.*, vol. 57, no. 3, pp. 1058–1067, 2009.

[25] K. Slavakis and S. Theodoridis, "Sliding window generalized Kernel affine projection algorithm using projection mappings," *EURASIP J. Adv. Signal Process.*, vol. 2008, Article ID 735351, 16 pp.

[26] W. Liu and J. C. Príncipe, "Kernel affine projection algorithms," *EURASIP J. Adv. Signal Process.*, vol. 2008, Article ID 784292, 12 pp.

[27] C. Richard, "Filtrage adaptatif non-linéaire par méthodes de gradient stochastique court-terme à noyau," in *Proc. Actes du 20e Colloque GRETSI sur le Traitement du Signal et des Images*, Louvain-la-Neuve, Belgium, 2005, pp. 41–44.

[28] W. Liu, P. P. Pokharel, and J. C. Príncipe, "The Kernel least-mean-square algorithm," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 543–554, 2008.

[29] 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 Trans. Signal Process.*, vol. 60, no. 5, pp. 2208–2222, 2012.

[30] C. Richard and J.-C. M. Bermudez, "Closed-form conditions for convergence of the Gaussian Kernel-least-mean-square algorithm," in *Proc. Asilomar*, 2012, pp. 1797–1801.

[31] B. Chen, S. Zhao, P. Zhu, and J. C. Príncipe, "Quantized Kernel least-mean-square algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 1, pp. 22–32, 2012.

[32] B. Chen, S. Zhao, S. Seth, and J. C. Príncipe, "Online efficient learning with quantized KLMS and L1 regularization," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2012, pp. 1–6.

[33] M. Yukawa, "Multikernel adaptive filtering," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4672–4682, 2012.

[34] W. Gao, J. Chen, C. Richard, J. Huang, and R. Flamary, "Kernel LMS algorithm with forward-backward splitting for dictionary learning," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2013, pp. 5735–5739.

[35] P. Bouboulis and S. Theodoridis, "Extension of Wirtinger's calculus to reproducing Kernel Hilbert spaces and the complex Kernel LMS," *IEEE Trans. Signal Process.*, vol. 59, no. 3, pp. 964–978, 2011.

[36] P. Bouboulis, S. Theodoridis, and M. Mavroforakis, "The augmented complex Kernel LMS," *IEEE Trans. Signal Process.*, vol. 60, no. 9, pp. 4962–4967, 2012.

[37] T. Paul and T. Ogunfunmi, "Analysis of the convergence behavior of the complex Gaussian Kernel LMS algorithm," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2012, pp. 2761–2764.

[38] F. A. Tobar and D. P. Mandic, "The quaternion Kernel least squares," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2013, pp. 6128–6132.

[39] J. Platt, "A resource-allocating network for function interpolation," *Neural Comput.*, vol. 3, no. 2, pp. 213–225, 1991.
[40] W. Liu, I. Park, and J. C. Príncipe, "An information theoretic approach of designing sparse Kernel adaptive filters," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1950–1961, 2009.
[41] K. Slavakis, P. Bouboulis, and S. Theodoridis, "Online learning in reproducing Kernel Hilbert spaces," in *E-Reference, Signal Processing*. Amsterdam, The Netherlands: Elsevier, 2013.
[42] Y. Chen, Y. Gu, and A. O. Hero, "Sparse LMS for system identification," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2009, pp. 3125–3128.
[43] K. Slavakis, Y. Kopsinis, and S. Theodoridis, "Adaptive algorithm for sparse system identification using projections onto weighted ℓ_1 balls," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2010, pp. 3742–3745.
[44] Y. Murakami, M. Yamagishi, M. Yukawa, and I. Yamada, "A sparse adaptive filtering using time-varying soft-thresholding techniques," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, 2010, pp. 3734–3737.
[45] E. J. Candès, Y. C. Eldar, D. Needell, and P. Randall, "Compressed sensing with coherent and redundant dictionaries," *Appl. Comput. Harmon. Anal.*, vol. 31, no. 1, pp. 59–73, 2011.
[46] F. Cucker and S. Smale, "On the mathematical foundations of learning," *Bull. Amer. Math. Soc.*, vol. 31, no. 1, pp. 1–49, 2001.
[47] J. Omura and T. Kailath, "Some useful probability distributions," Stanford Electronics Laboratories, Stanford Univ., Stanford, CA, USA, Tech. Rep. 7050-6, 1965.
[48] J. Minkoff, "Comment: On the unnecessary assumption of statistical independence between reference signal and filter weights in feedforward adaptive systems," *IEEE Trans. Signal Process.*, vol. 49, no. 5, p. 1109, 2001.
[49] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 3–27, 1990.
[50] D. P. Mandic, "A generalized normalized gradient descent algorithm," *IEEE Signal Process. Lett.*, vol. 2, pp. 115–118, 2004.
[51] J. Vörös, "Modeling and identification of Wiener systems with two-segment nonlinearities," *IEEE Trans. Control Syst. Technol.*, vol. 11, no. 2, pp. 253–257, 2003.
[52] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. New York, NY, USA: Springer, 2011.
[53] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.
[54] J. Duchi and Y. Singer, "Efficient online and batch learning using forward backward splitting," *J. Mach. Learn. Res.*, vol. 10, pp. 2899–2934, 2009.
[55] H. Zou, "The adaptive lasso and its oracle properties," *J. Amer. Statist. Assoc.*, vol. 101, no. 476, pp. 1418–1429, 2006.
[56] E. J. Candès, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted ℓ_1 minimization," *J. Fourier Anal. Appl.*, vol. 14, no. 5, pp. 877–905, 2008.

Wei Gao (S'11) received the M.S. degree in information and communication engineering from Northwestern Polytechnical University (NPU), in Xi'an, China, in 2010. He is currently pursuing his Ph.D. degree at NPU since 2010, and simultaneously preparing a Ph.D. degree at University of Nice Sophia Antipolis (UNS), Nice, France, since 2012. His current research interests include nonlinear adaptive filtering, adaptive noise cancellation and array signal processing.

Jie Chen (S'12–M'14) was born in Xi'an, China, in 1984. He received the B.S. degree in information and telecommunication engineering in 2006 from the Xi'an Jiaotong University, Xi'an, and the Dipl.-Ing. and the M.S. degrees in information and telecommunication engineering in 2009 from the University of Technology of Troyes (UTT), Troyes, France, and from the Xi'an Jiaotong University, respectively. In 2013, he received the Ph.D. degree in systems optimization and security from the UTT.

From April 2013 to March 2014, he was a Postdoctoral Researcher at the Côte d'Azur Observatory, University of Nice Sophia Antipolis, Nice, France. Since April 2014, he works as a Postdoctoral Researcher with the Department of Electrical Engineering and Computer Science of University of Michigan, Ann Arbor, USA.

His current research interests include adaptive signal processing, kernel methods, hyperspectral image analysis, distributed optimization, and supervised and unsupervised learning. He is a reviewer for several journals, including IEEE TRANSACTIONS ON IMAGE PROCESSING, IEEE TRANSACTIONS ON GEOSCIENCE AND REMOTE SENSING and *Elsevier Signal Processing*.

Cédric Richard (S'98–M'01–SM'07) was born January 24, 1970 in Sarrebourg, France. He received the Dipl.-Ing. and the M.S. degrees in 1994 and the Ph.D. degree in 1998 from the University of Technology of Compiègne, France, all in electrical and computer engineering. From 1999 to 2003, he was an Associate Professor at the University of Technology of Troyes (UTT), France. From 2003 to 2009, he was a Full Professor at UTT. Since september 2009, he is a Full Professor in the Lagrange Laboratory (University of Nice Sophia Antipolis, CNRS, Observatoire de la Côte d'Azur). In winter 2009 and 2014, and autumns 2010, 2011 and 2013, he was a Visiting Researcher with the Department of Electrical Engineering, Federal University of Santa Catarina (UFSC), Florianopolis, Brazil. He is a junior member of the Institut Universitaire de France since October 2010.

His current research interests include statistical signal processing and machine learning.

Cédric Richard is the author of over 220 papers. He was the General Chair of the XXIth francophone conference GRETSI on Signal and Image Processing that was held in Troyes, France, in 2007, and of the IEEE Statistical Signal Processing Workshop (IEEE SSP'11) that was held in Nice, France, in 2011. He will be the Technical Chair of EUSIPCO 2015. Since 2005, he is a member of GRETSI association board and of the EURASIP society, and Senior Member of the IEEE. In 2006–2010, he served as an associate editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING. Actually, he serves as an Associate Editor of *Elsevier Signal Processing*, and of the IEEE SIGNAL PROCESSING LETTERS. He is an EURASIP liaison local officer. He is a member of the Signal Processing Theory and Methods (SPTM TC) Technical Committee, and of the Machine Learning for Signal Processing (MLSP TC) Technical Committee, of the IEEE Signal Processing Society.

Paul Honeine and and Cédric Richard won the Best Paper Award for "Solving the preimage problem in kernel machines: a direct method" at the 2009 IEEE International Workshop on Machine Learning for Signal Processing.

Jianguo Huang (SM'94) received the B.E. degree in acoustics and electronic engineering from Northwestern Polytechnical University (NPU), China, in 1967. He conducted his Master study for signal processing in Peking University, from 1979 to 1980. From 1985 to 1988, he was a Visiting Researcher for modern spectral estimation in University of Rhode Island, USA. In 1998, as a Visiting Professor he conducted the joint research on high resolution parameter estimation in State University of New York, Stony Brook, USA.

He is currently a Professor of signal processing and wireless communication, Director of the Institute of Oceanic Science and Information Processing in NPU, adjunct professor of Shanghai Jiaotong University, the Fellow of Chinese Society of Acoustics. He was also the former Dean of the College of Marine Engineering and Director of State Key Laboratory of Underwater Information Processing and Control in NPU. He is the author of more than 520 papers and five books. His general research interests include modern signal processing, array signal processing, and wireless and underwater acoustic communication theory and application.

Mr. Huang has been in charge of more than 40 projects of national and ministries science foundation. For the achievements of research in theory and applications, he obtained 20 awards from nation, ministries and province. He was recognized as the "2008 Chinese Scientist of the Year". Prof. Huang was invited by the universities and companies to deliver the lectures in USA, U.K., Canada, Australia, Singapore, Japan and Hong Kong. He is the Chairman of IEEE Xi'an Section of China, honorary Chair of ChinaSIP2013, and one of the founders and a member of the Steering Committee of IEEE ChinaSIP. He also served as General or TPC Chair/Co-Chair in many international conferences, including IEEE ICSPCC, IEEE TENCON2013, IEEE ICCSN2011, IEEE ICIEA2009 and so on.