Online Prediction of Time Series Data With Kernels

Cédric Richard, *Senior Member, IEEE*, José Carlos M. Bermudez, *Senior Member, IEEE*, and Paul Honeine, *Member, IEEE*

***Abstract*—Kernel-based algorithms have been a topic of considerable interest in the machine learning community over the last ten years. Their attractiveness resides in their elegant treatment of nonlinear problems. They have been successfully applied to pattern recognition, regression and density estimation. A common characteristic of kernel-based methods is that they deal with kernel expansions whose number of terms equals the number of input data, making them unsuitable for online applications. Recently, several solutions have been proposed to circumvent this computational burden in time series prediction problems. Nevertheless, most of them require excessively elaborate and costly operations. In this paper, we investigate a new model reduction criterion that makes computationally demanding sparsification procedures unnecessary. The increase in the number of variables is controlled by the coherence parameter, a fundamental quantity that characterizes the behavior of dictionaries in sparse approximation problems. We incorporate the coherence criterion into a new kernel-based affine projection algorithm for time series prediction. We also derive the kernel-based normalized LMS algorithm as a particular case. Finally, experiments are conducted to compare our approach to existing methods.**

***Index Terms*—Adaptive filters, machine learning, nonlinear systems, pattern recognition.**

I. Introduction

DYNAMIC system modeling has played a crucial role in the development of techniques for stationary and nonstationary signal processing. 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. See extensive bibliography [1] devoted to the theory of nonlinear systems. Unlike the case of linear systems which can be uniquely identified by their impulse response, there is a wide variety of representations to characterize nonlinear systems, ranging from higher-order statistics, e.g., [2], [3], to series expansion methods, e.g., [4], [5]. Two main types of nonlinear models have been extensively studied over the years: polynomial filters, usually called Volterra series based filters [6], and neural networks [7]. The Volterra filters can model a large class of nonlinear systems. They 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. One of their primary disadvantages is the considerable number of parameters to estimate, which goes up exponentially as the order of the nonlinearity increases. With their parallel structure, neural networks represent the ultimate development of black box modeling [8]. They are proven to be universal approximators under suitable conditions, thus, providing the means to capture information in data that is difficult to identify using other techniques [9]. 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.

Since the pioneering works of Aronszajin [10], Aizerman *et al.* [11], Kimeldorf and Wahba [12], [13], and Duttweiler and Kailath [14], function approximation methods based on reproducing kernel Hilbert spaces (RKHS) have gained wide popularity [15]. Recent developments in kernel-based methods related to regression include, most prominently, support vector regression [16], [17]. A key property behind such algorithms is that the only operations they require is the evaluation of inner products between pairs of the input vectors. Replacing inner products with a Mercer kernel provides an efficient way to implicitly map the data into a high, even infinite, dimensional RKHS and apply the original algorithm in this space. Calculations are then carried out without making direct reference to the nonlinear mapping of input vectors. A common characteristic in kernel-based methods is that they deal with matrices whose size equals the number of data, making them unsuitable for online applications. Several attempts have been made recently to circumvent this computational burden. A gradient descent method is applied in [18] and [19], while a RLS-like procedure is used in [20] to update the model parameters. Each one is associated with a sparsification procedure based on the matrix inversion lemma, which limits the increase in the number of terms by including only kernels that significantly reduce the approximation error. These processes have reduced the computational burden of the traditional approaches. Nevertheless, they still require elaborate and costly operations, that limits their applicability in real-time systems.

In this paper, we investigate a new model reduction criterion that renders computationally demanding sparsification procedures unnecessary. The increase in the number of variables is controlled by the coherence parameter, a fundamental quantity that characterizes the behavior of dictionaries in sparse approximation problems. We associate the coherence criterion with a new kernel-based algorithm for time series prediction, called kernel affine projection (KAP) algorithm. We also derive the

Manuscript received March 19, 2008; revised October 24, 2008. First published November 21, 2008; current version published February 13, 2009. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Manuel Davy.

C. Richard and P. Honeine are with the Institut Charles Delaunay (FRE CNRS 2848), Laboratoire LM2S, Université de technologie de Troyes, 10010 Troyes, France (e-mail: cedric.richard@utt.fr; paul.honeine@utt.fr).

J. Bermudez is with the Department of Electrical Engineering, Federal University of Santa Catarina 88040-900, Florianópolis, SC, Brazil (e-mail: j.bermudez@ieee.org).

Digital Object Identifier 10.1109/TSP.2008.2009895

kernel normalized LMS (KNLMS) algorithm as a particular case. The paper is organized as follows. In the first part, we briefly review some basic principles of nonlinear regression in RKHS. Next we show how to use the coherence parameter as an alternative criterion for model sparsification, and we derive its main properties. We then incorporate it into our KAP algorithm, which includes as particular case the KNLMS algorithm. Finally, a set of experiments illustrate the effectiveness of the proposed method compared to other existing approaches.

II. Principles of Nonlinear Regression in RKHS

A possible way to extend the scope of linear models to nonlinear processing is to map the input data $\boldsymbol{u}_i$ into a high-dimensional space using a nonlinear function $\boldsymbol{\varphi}(\cdot)$, and apply linear modeling techniques to the transformed data $\boldsymbol{\varphi}(\boldsymbol{u}_i)$. The model coefficients are then determined as the solution of the normal equations written for the nonlinearly transformed input data. Clearly, this basic strategy may fail when the image of $\boldsymbol{\varphi}(\cdot)$ is a very high, or even infinite, dimensional space. Kernel-based methods that lead manageable dimensions have been recently proposed for applications in classification and regression problems. Well-known examples can be found in [15] and [21]. This paper exploits the central idea of this research area, known as the *kernel trick*, to investigate new nonlinear algorithms for online prediction of time series. Next section briefly reviews the main definitions and properties related to reproducing kernel Hilbert spaces [10] and Mercer kernels [22].

A. RKHS and Mercer Kernels

Let $\mathcal{H}$ denote a Hilbert space of real-valued functions $\psi(\cdot)$ on a compact $\mathcal{U} \subset \mathbb{R}^\ell$, and let $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ be the inner product in $\mathcal{H}$. Suppose that the evaluation functional L_u defined by $L_u[\psi] \triangleq \psi(\boldsymbol{u})$ is linear with respect to $\psi(\cdot)$ and bounded, for all $\boldsymbol{u}$ in $\mathcal{U}$. By virtue of the Riesz representation theorem, there exists a unique positive definite function $\boldsymbol{u}_i \mapsto \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$ in $\mathcal{H}$, denoted by $\kappa(\cdot, \boldsymbol{u}_j)$ and called *representer of evaluation* at $\boldsymbol{u}_j$, which satisfies [10]

$$\psi(\boldsymbol{u}_j) = \langle \psi(\cdot), \kappa(\cdot, \boldsymbol{u}_j) \rangle_{\mathcal{H}}, \quad \forall \psi \in \mathcal{H} \tag{1}$$

for every fixed $\boldsymbol{u}_j \in \mathcal{U}$. A proof of this may be found in [10]. Replacing $\psi(\cdot)$ by $\kappa(\cdot, \boldsymbol{u}_i)$ in (1) yields

$$\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \langle \kappa(\cdot, \boldsymbol{u}_i), \kappa(\cdot, \boldsymbol{u}_j) \rangle_{\mathcal{H}} \tag{2}$$

for all $\boldsymbol{u}_i, \boldsymbol{u}_j \in \mathcal{U}$. Equation (2) is the origin of the now generic term *reproducing kernel* to refer to $\kappa(\cdot, \cdot)$. Note that $\mathcal{H}$ can be restricted to the span of $\{\kappa(\cdot, \boldsymbol{u}) : \boldsymbol{u} \in \mathcal{U}\}$ because, according to (1), nothing outside this set affects $\psi(\cdot)$ evaluated at any point of $\mathcal{U}$. Denoting by $\boldsymbol{\varphi}(\cdot)$ the map that assigns to each input $\boldsymbol{u}$ the kernel function $\kappa(\cdot, \boldsymbol{u})$, (2) implies that $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \langle \boldsymbol{\varphi}(\boldsymbol{u}_i), \boldsymbol{\varphi}(\boldsymbol{u}_j) \rangle_{\mathcal{H}}$. The kernel then evaluates the inner product of any pair of elements of $\mathcal{U}$ mapped to $\mathcal{H}$ without any explicit knowledge of either $\boldsymbol{\varphi}(\cdot)$ or $\mathcal{H}$. This key idea is known as the *kernel trick*.

Classic examples of kernels are the radially Gaussian kernel $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2 / 2\beta_0^2)$, and the Laplacian kernel $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\| / \beta_0)$, with $\beta_0 \geq 0$ the kernel bandwidth. Another example which deserves attention in signal processing is the qth degree polynomial kernel defined as $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = (\eta_0 + \boldsymbol{u}_i^t \boldsymbol{u}_j)^q$, with $\eta_0 \geq 0$ and $q \in \mathbb{N}^*$. The nonlinear function $\boldsymbol{\varphi}(\cdot)$ related to the latter transforms every observation $\boldsymbol{u}_i$ into a vector $\boldsymbol{\varphi}(\boldsymbol{u}_i)$, in which each component is proportional to a monomial of the form $(u_{i,1})^{k_1} (u_{i,2})^{k_2} \ldots (u_{i,p})^{k_p}$ for every set of exponents satisfying $0 \leq \sum_{r=1}^{p} k_r \leq q$. For details, see [23], [24], and references therein. The models of interest then correspond to qth degree Volterra series representations.

B. Nonlinear Regression With Mercer Kernels

The kernel trick has been widely used to transform linear algorithms expressed only in terms of inner products into nonlinear ones. Examples are the nonlinear extensions to the principal components analysis [25] and the Fisher discriminant analysis [26], [27]. Recent work has been focussed on kernel-based online prediction of time series [18]–[20], the topic of this article. Let $\kappa : \mathcal{U} \times \mathcal{U} \to \mathbb{R}$ be a kernel, and let $\mathcal{H}$ be the RKHS associated with it. Considering the least-squares approach, the problem is to determine a function $\psi(\cdot)$ of $\mathcal{H}$ that minimizes the sum of n squared errors between samples d_i of the desired response and the corresponding model output samples $\psi(\boldsymbol{u}_i) = \langle \psi(\cdot), \kappa(\cdot, \boldsymbol{u}_i) \rangle_{\mathcal{H}}$, namely

$$\min_{\psi \in \mathcal{H}} \sum_{i=1}^{n} |d_i - \psi(\boldsymbol{u}_i)|^2 . \tag{3}$$

By virtue of the representer theorem [12], [28], the function $\psi(\cdot)$ of $\mathcal{H}$ minimizing (3) can be written as a kernel expansion in terms of available data

$$\psi(\cdot) = \sum_{j=1}^{n} \alpha_j \kappa(\cdot, \boldsymbol{u}_j). \tag{4}$$

It can be shown that (3) becomes $\min_{\boldsymbol{\alpha}} \|\boldsymbol{d} - \boldsymbol{K}\boldsymbol{\alpha}\|^2$, where $\boldsymbol{K}$ is the Gram matrix whose (i, j)th entry is $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)$. The solution vector $\boldsymbol{\alpha}$ is found by solving the n-by-n linear system of equations $\boldsymbol{K}\boldsymbol{\alpha} = \boldsymbol{d}$.

III. A New Model Reduction Method

Online prediction of time series data raises the question of how to process an increasing amount of observations and update the model (4) as new data is collected. We focus on fixed-size models of the form

$$\psi_n(\cdot) = \sum_{j=1}^{m} \alpha_j \kappa\left(\cdot, \boldsymbol{u}_{\omega_j}\right) \tag{5}$$

at any time step n, where the ω_j's form an m-element subset $\mathcal{J}_n$ of $\{1, \ldots, n\}$. We call $\{\kappa(\cdot, \boldsymbol{u}_{\omega_j})\}_{j=1}^{m}$ the dictionary, and m the order of the kernel expansion by analogy with linear transversal filters. Online identification of kernel-based models generally relies on a two-stage process at each iteration: a model order control step that inserts and removes kernel functions from the dictionary, and a parameter update step.

A. A Brief Review of Sparsification Rules

Discarding a kernel function from the model expansion (5) may degrade its performance. Sparsification rules aim at identi-

fying kernel functions whose removal is expected to have negligible effect on the quality of the model. An extensive literature addressing this issue in batch and online modes exists, see, e.g. [29] and references therein. In particular, much attention has been recently focused on least-squares support vector machines since they suffer from the loss of sparsity due to the use of a quadratic loss function [17]. In batch modes, this problem was addressed by using pruning [30], [31] and fixed-size approaches [17], [32], [33]. Truncation and approximation processes were considered in online scenarios [29].

The most informative sparsification criteria use approximate linear dependence conditions to evaluate whether the contribution of a candidate kernel function can be distributed over the elements of the dictionary by adjusting their multipliers. In [34], determination of the kernel function which is best approximated by the others is carried out by an eigendecomposition of the Gram matrix. This process is not appropriate for online applications since its complexity, at each time step, is cubic in the size m of the dictionary. In [20], the kernel function $\kappa(\cdot,\boldsymbol{u}_n)$ is inserted at time step n into the dictionary if the following condition is satisfied

$$\min_{\gamma}\left\|\kappa(\cdot,\boldsymbol{u}_n)-\sum_{\omega_j\in\mathcal{J}_{n-1}}\gamma_j\kappa\left(\cdot,\boldsymbol{u}_{\omega_j}\right)\right\|_{\mathcal{H}}^2\geq\nu \tag{6}$$

where κ is a unit-norm kernel,[1] that is, $\kappa(\boldsymbol{u}_k,\boldsymbol{u}_k)=1$ for all $\boldsymbol{u}_k$. The threshold ν determines the level of sparsity of the model. Note that (6) ensures the linear independence of the elements of the dictionary. A similar criterion is used in [18] and [19], but in a different form. After updating the model parameters, a complementary pruning process is executed to limit the increase in the model order in [19]. It estimates the error induced in $\psi_n(\cdot)$ by the removal of each kernel and discards those kernels found to have the smallest contribution. A major criticism that can be made of rule (6) is that it leads to elaborate and costly operations with quadratic complexity in the cardinality m of the dictionary. In [18] and [19], the model reduction step is computationally more expensive than the parameter update step, the latter being a stochastic gradient descent with linear complexity in m. In [20], the authors focus their study on a parameter update step of the RLS type with quadratic complexity in m. To reduce the overall computational effort, the parameter update and the model reduction steps share intermediate results of calculations. This excludes very useful and popular online regression techniques.

B. Redundant Dictionaries, Coherence and Babel Function

Coherence is a fundamental parameter to characterize a dictionary in linear sparse approximation problems [35]. It was introduced as a quantity of heuristic interest by Mallat and Zhang for Matching Pursuit [36]. The first formal developments were described in [37], and enriched for Basis Pursuit in [38] and [39]. In our kernel-based context, we propose to define the coherence parameter as

$$\mu=\max_{i\neq j}\left|\left\langle\kappa\left(\cdot,\boldsymbol{u}_{\omega_i}\right),\kappa\left(\cdot,\boldsymbol{u}_{\omega_j}\right)\right\rangle_{\mathcal{H}}\right|=\max_{i\neq j}\left|\kappa\left(\boldsymbol{u}_{\omega_i},\boldsymbol{u}_{\omega_j}\right)\right| \tag{7}$$

where κ is a unit-norm kernel (see footnote 1). The parameter μ is then the largest absolute value of the off-diagonal entries in the Gram matrix. It reflects the largest cross correlations in the dictionary. Consequently, it is equal to zero for every orthonormal basis. A dictionary is said to be incoherent when μ is small.

Now, consider the Babel function given by

$$\mu_1(m)=\max_{\omega_0}\max_{\substack{\omega_j\in\mathcal{J}\\ \omega_j\neq\omega_0}}\sum_{j=1}^{m}\left|\kappa\left(\boldsymbol{u}_{\omega_0},\boldsymbol{u}_{\omega_j}\right)\right| \tag{8}$$

where $\mathcal{J}$ is a set of m indices. Function $\mu_1(m)$ is defined as the maximum total coherence between a fixed kernel function $\kappa(\cdot,\boldsymbol{u}_{\omega_0})$ and a subset of m other functions $\kappa(\cdot,\boldsymbol{u}_{\omega_j})$ of the dictionary. It provides a more in-depth description of a dictionary. We note that $\mu_1(m)\leq m\mu$ for a dictionary with coherence μ, as $|\kappa(\boldsymbol{u}_{\omega_0},\boldsymbol{u}_{\omega_j})|\leq\mu$ for any distinct ω_0 and ω_j in this case. The following proposition establishes a useful sufficient condition for a dictionary of kernel functions to be linearly independent.

Proposition 1: Let $\kappa(\cdot,\boldsymbol{u}_1),\ldots,\kappa(\cdot,\boldsymbol{u}_m)$ be an arbitrary set of m kernel functions from a dictionary, and let $\mu_1(m)$ be the Babel function evaluated for this set. If $\mu_1(m-1)<1$, then this set is linearly independent.

Proof: Consider any linear combination $\sum_{i=1}^{m}\alpha_i\kappa(\cdot,\boldsymbol{u}_i)$. We have

$$\left\|\sum_{i=1}^{m}\alpha_i\kappa(\cdot,\boldsymbol{u}_i)\right\|_{\mathcal{H}}^2=\boldsymbol{\alpha}^t\boldsymbol{K}\boldsymbol{\alpha}\geq\lambda_{\min}\|\boldsymbol{\alpha}\|^2\geq 0$$

where $\lambda_{\min}$ is the smallest eigenvalue of the Gram matrix $\boldsymbol{K}$. According to the Geršgorin disk theorem [40], every eigenvalue of $\boldsymbol{K}$ lies in the union of the m disks $\{\lambda:|\lambda-\kappa(\boldsymbol{u}_i,\boldsymbol{u}_i)|\leq r_i\}$, each centered on the diagonal element $\kappa(\boldsymbol{u}_i,\boldsymbol{u}_i)$ of $\boldsymbol{K}$ and with radii $r_i=\sum_{j\neq i}|\kappa(\boldsymbol{u}_i,\boldsymbol{u}_j)|$ for all $1\leq i\leq m$. The normalization of the kernel and the definition of the Babel function yield $|\lambda-1|\leq\mu_1(m-1)$. The result follows directly since $\lambda_{\min}>0$ if $\mu_1(m-1)<1$. ∎

If computation of $\mu_1(m-1)$ becomes too expensive, the simpler but somewhat more restrictive sufficient condition $(m-1)\mu<1$ can be used, since $\mu_1(m-1)\leq(m-1)\mu$. The results above show that the coherence coefficient (7) provides valuable information on the linear independence of the kernel functions of a dictionary at low computational cost. In the following we show how to use it for sparsification of kernel expansions as an efficient alternative to the approximate linear condition (6).

C. The Coherence-Based Sparsification Rule

Typical sparsification methods use approximate linear dependence conditions to evaluate whether, at each time step n, the new candidate kernel function $\kappa(\cdot,\boldsymbol{u}_n)$ can be reasonably well

[1]Replace $\kappa(\cdot,\boldsymbol{u}_k)$ with $\kappa(\cdot,\boldsymbol{u}_k)/\sqrt{\kappa(\boldsymbol{u}_k,\boldsymbol{u}_k)}$ in (6) if $\kappa(\cdot,\boldsymbol{u}_k)$ is not unit-norm.

represented by a combination of the kernel functions of the dictionary. If not, it is added to the dictionary. To avoid the computational complexity inherent to these methods, we suggest inserting $\kappa(\cdot,\boldsymbol{u}_n)$ into the dictionary provided that its coherence remains below a given threshold μ_0, namely

$$\max_{\omega_j\in\mathcal{J}_{n-1}} \left|\kappa\left(\boldsymbol{u}_n,\boldsymbol{u}_{\omega_j}\right)\right| \leq \mu_0 \tag{9}$$

where μ_0 is a parameter in [0,1[determining both the level of sparsity and the coherence of the dictionary. We shall now show that, under a reasonable condition on $\mathcal{U}$, the dimension of the dictionary determined under rule (9) remains finite as n goes to infinity.

Proposition 2: Let $\mathcal{U}$ be a compact subspace of a Banach space, and $\kappa : \mathcal{U}\times\mathcal{U} \rightarrow \mathbb{R}$ be a Mercer kernel. Then, the dimension of the dictionary determined under the sparsification rule (9) with $0 \leq \mu_0 < 1$ is finite for any sequence $\{\boldsymbol{u}_i\}_{i=1}^{\infty}$.

Proof: From the compactness of $\mathcal{U}$ and continuity of $\kappa(\cdot,\boldsymbol{u})$, we know that $\{\kappa(\cdot,\boldsymbol{u}) : \boldsymbol{u} \in \mathcal{U}\}$ is compact. This implies that a finite open cover of ℓ_2-balls of this set exists. We observe that, under (9), any two kernel functions $\kappa(\cdot,\boldsymbol{u}_{\omega_i})$ and $\kappa(\cdot,\boldsymbol{u}_{\omega_j})$ in the dictionary verify $\|\kappa(\cdot,\boldsymbol{u}_{\omega_i})-\kappa(\cdot,\boldsymbol{u}_{\omega_j})\|^2_{\mathcal{H}} = 2-2\kappa(\boldsymbol{u}_{\omega_i},\boldsymbol{u}_{\omega_j}) \geq 2-2\mu_0$. Then, the number of such balls is finite. ∎

The above proposition implies that the computational cost per time-step of algorithms implementing the strategy (9) becomes independent of time after a transient period. After such period, the computational cost depends only on the cardinality m of the final dictionary, which is a function of the threshold μ_0. For instance, we set μ_0 in the numerical experiments presented in Section V so that m never exceeds a few tens. Since the proposed sparsification rule is an alternative to the approximate condition (6), it is of interest to establish a connection between that condition and rule (9). We do this in the following proposition.

Proposition 3: Let $\kappa(\cdot,\boldsymbol{u}_{\omega_1}),\ldots,\kappa(\cdot,\boldsymbol{u}_{\omega_m})$ be m kernel functions selected by the coherence-based rule (9). If $(m-1)\mu_0 < 1$, then the norm of the projection of $\kappa(\cdot,\boldsymbol{u}_{\omega_m})$ onto the span of the other $(m-1)$ kernel functions is less than or equal to $\sqrt{(m-1)\mu_0^2/(1-(m-1)\mu_0)}$.

Proof: Let $\mathcal{S}$ denote the span of $\kappa(\cdot,\boldsymbol{u}_{\omega_1}),\ldots,\kappa(\cdot,\boldsymbol{u}_{\omega_{m-1}})$ and let $P_{\mathcal{S}}\kappa(\cdot,\boldsymbol{u}_{\omega_m})$ be the projection of the kernel function $\kappa(\cdot,\boldsymbol{u}_{\omega_m})$ onto $\mathcal{S}$. The norm of $P_{\mathcal{S}}\kappa(\cdot,\boldsymbol{u}_{\omega_m})$ is the maximum, over all the unit functions $\psi(\cdot)$ of $\mathcal{S}$, of the inner product $\langle\kappa(\cdot,\boldsymbol{u}_{\omega_m}),\psi(\cdot)\rangle_{\mathcal{H}}$. Writing $\psi(\cdot) = \sum_{i=1}^{m-1}\alpha_i\kappa(\cdot,\boldsymbol{u}_{\omega_i})/\|\sum_{i=1}^{m-1}\alpha_i\kappa(\cdot,\boldsymbol{u}_{\omega_i})\|_{\mathcal{H}}$, the problem can be formally stated as follows:

$$\begin{aligned}
&\|P_{\mathcal{S}}\kappa\left(\cdot,\boldsymbol{u}_{\omega_m}\right)\|_{\mathcal{H}} \\
&\quad= \max_{\alpha} \frac{\left\langle\sum_{i=1}^{m-1}\alpha_i\kappa\left(\cdot,\boldsymbol{u}_{\omega_i}\right),\kappa\left(\cdot,\boldsymbol{u}_{\omega_m}\right)\right\rangle_{\mathcal{H}}}{\left\|\sum_{i=1}^{m-1}\alpha_i\kappa\left(\cdot,\boldsymbol{u}_{\omega_i}\right)\right\|_{\mathcal{H}}} \qquad (10)\\
&\quad= \max_{\alpha} \frac{\left(\sum_{i=1}^{m-1}\alpha_i\kappa\left(\boldsymbol{u}_{\omega_i},\boldsymbol{u}_{\omega_m}\right)\right)}{\left\|\sum_{i=1}^{m-1}\alpha_i\kappa\left(\cdot,\boldsymbol{u}_{\omega_i}\right)\right\|_{\mathcal{H}}}. \qquad (11)
\end{aligned}$$

On the one hand, the numerator of this expression can be upper bounded as follows:

$$\begin{aligned}
\left[\sum_{i=1}^{m-1}\alpha_i\kappa\left(\boldsymbol{u}_{\omega_i},\boldsymbol{u}_{\omega_m}\right)\right]^2 &\leq \left[\sum_{i=1}^{m-1}\mu_0|\alpha_i|\right]^2 \\
&\leq (m-1)\mu_0^2\|\boldsymbol{\alpha}\|^2 \qquad (12)
\end{aligned}$$

where the last inequality follows from the Cauchy-Schwartz inequality. On the other hand, the denominator in (11) can be lower bounded as follows:

$$\begin{aligned}
\left\|\sum_{i=1}^{m-1}\alpha_i\kappa\left(\cdot,\boldsymbol{u}_{\omega_i}\right)\right\|^2_{\mathcal{H}} &= \boldsymbol{\alpha}^t\boldsymbol{K}\boldsymbol{\alpha} \geq \lambda_{\min}\|\boldsymbol{\alpha}\|^2 \\
&\geq \left[1-(m-2)\mu_0\right]\|\boldsymbol{\alpha}\|^2 \qquad (13)
\end{aligned}$$

where $\boldsymbol{K}$ denotes here the Gram matrix of the $(m-1)$ kernel functions $\kappa(\cdot,\boldsymbol{u}_{\omega_i})$. The last inequality follows from the Geršgorin disk theorem [40]. Finally, combining inequalities (12) and (13) with (11) yields

$$\|P_{\mathcal{S}}\kappa\left(\cdot,\boldsymbol{u}_{\omega_m}\right)\|_{\mathcal{H}} \leq \sqrt{\frac{(m-1)\mu_0^2}{1-(m-2)\mu_0}}. \tag{14}$$

This bound is valid and non-trivial if it lies in the interval [0,1[, that is, if and only if $(m-1)\mu_0 < 1$. This is also the sufficient condition stated in Proposition 1 for the $\kappa(\cdot,\boldsymbol{u}_{\omega_j})$'s to be linearly independent. ∎

The projection of $\kappa(\cdot,\boldsymbol{u}_{\omega_m})$ onto the space spanned by the $(m-1)$ previously selected kernel functions results in a squared error $\|(I-P_{\mathcal{S}})\kappa(\cdot,\boldsymbol{u}_{\omega_m})\|^2_{\mathcal{H}}$. From Proposition 3, we deduce that

$$\begin{aligned}
&\|(I-P_{\mathcal{S}})\kappa\left(\cdot,\boldsymbol{u}_{\omega_m}\right)\|^2_{\mathcal{H}} \\
&\quad= \|\kappa\left(\cdot,\boldsymbol{u}_{\omega_m}\right)\|^2_{\mathcal{H}} - \|P_{\mathcal{S}}\kappa\left(\cdot,\boldsymbol{u}_{\omega_m}\right)\|^2_{\mathcal{H}} \qquad (15)\\
&\quad\geq 1-\frac{(m-1)\mu_0^2}{1-(m-2)\mu_0} \qquad (16)
\end{aligned}$$

under $(m-1)\mu_0 < 1$, which ensures that the lower bound lies in the interval]0,1]. As expected, the smaller m and μ_0, the larger the squared error in the approximation of any dictionary element by a linear combination of the others. We conclude that the coherence-based rule (9) implicitly specifies a lower bound on the squared error $\|(I-P_{\mathcal{S}})\kappa(\cdot,\boldsymbol{u}_{\omega_m})\|^2_{\mathcal{H}}$ via μ_0, a mechanism which is explicitly governed by ν in the approximate linear condition (6). Both approaches can then generate linearly independent sets of kernel functions, a constraint that will be ignored in what follows. A major advantage of the coherence-based rule is that it is simpler and far less time consuming than (6). At each time-step, its computational complexity is only linear in the dictionary size m, whereas (6) has at least quadratic complexity even when computed recursively.

It is also of interest to establish a connection between the coherence-based rule and quadratic Renyi entropy. This measure, which quantifies the amount of disorder in a system, is defined

as follows: $H_R = -\log \int p(\boldsymbol{u})^2 d\boldsymbol{u}$ with p a probability density function. Consider first the Parzen density estimate

$$\hat{p}(\boldsymbol{u}) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{(\sqrt{\pi}\beta_0)^{\ell}} \exp\left(-\|\boldsymbol{u}\|^2/\beta_0^2\right) \tag{17}$$

based on the Gaussian window. By the convolution theorem applied to Gaussian distributions, we have

$$H_R \approx -\log \int \hat{p}(\boldsymbol{u})^2 d\boldsymbol{u} = -\log\left[\frac{1}{m^2}\sum_{i,j=1}^{m} \frac{\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)}{(2\pi\beta_0^2)^{\ell/2}}\right] \tag{18}$$

where $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2/2\beta_0^2)$ denotes the Gaussian kernel. The above example simply shows that the sum of the entries of the Gram matrix characterizes the diversity of the dictionary of kernel functions [41]. In [17], this was used as a criterion in a selection mechanism with fixed-size least-squares support vector machines. We observe in (18) that the coherence-based rule (9) ensures that

$$H_R \geq \log\left(2\pi\beta_0^2\right)^{\ell/2} - \log\left[\frac{1+(m-1)\mu_0}{m}\right]. \tag{19}$$

As expected, the lower bound on H_R increases as μ_0 decreases and m increases. In a more general way, since the integral $\int \hat{p}(\boldsymbol{u})^2 d\boldsymbol{u}$ also defines the squared norm $\|\hat{p}\|_{\mathcal{H}}^2$ of the functional form $\hat{p}(\cdot) = (1/m)\sum_{i=1}^{m} \kappa(\cdot, \boldsymbol{u}_i)$, it was observed in [41] that

$$H_R \approx -\log \|\hat{p}\|_{\mathcal{H}}^2 = -\log\left[\frac{1}{m^2}\sum_{i,j=1}^{m} \kappa(\boldsymbol{u}_i, \boldsymbol{u}_j)\right]. \tag{20}$$

In the case where κ is not a unit-norm kernel, remember that $\kappa(\cdot, \boldsymbol{u}_k)$ must be replaced by $\kappa(\cdot, \boldsymbol{u}_k)/\sqrt{\kappa(\boldsymbol{u}_k, \boldsymbol{u}_k)}$ in the coherence-based rule (9). Assuming that $\kappa(\boldsymbol{u}_k, \boldsymbol{u}_k) = \zeta$ for all k, (20) leads to

$$H_R \geq \log\left(\frac{1}{\zeta}\right) - \log\left[\frac{1+(m-1)\mu_0}{m}\right]. \tag{21}$$

Note that this bound, which depends on the norm of kernel functions, increases as μ_0 decreases or m increases. This result emphasizes the usefulness of coherence to accurately characterize the diversity of kernel functions in a dictionary. In the next section, we use this criterion to derive a new kernel-based algorithm for time series prediction, called kernel-based affine projection (KAP) algorithm.

IV. A Kernel-Based Affine Projection Algorithm With Order-Update Mechanism

Let $\hat{\psi}_n(\cdot)$ denote the mth-order model at time step n, with $m \leq n$. Then

$$\hat{\psi}_n(\cdot) = \sum_{j=1}^{m} \hat{\alpha}_{n,j}\kappa\left(\cdot, \boldsymbol{u}_{\omega_j}\right) \tag{22}$$

where the $\kappa(\cdot, \boldsymbol{u}_{\omega_j})$s form a μ_0-coherent dictionary determined under rule (9). In accordance with the least-squares problem described in Section II-B, the optimal $\hat{\boldsymbol{\alpha}}_n$ solves $\min_{\boldsymbol{\alpha}} \|\boldsymbol{d}_n - \boldsymbol{H}_n\boldsymbol{\alpha}\|^2$ where $\boldsymbol{H}_n$ denotes the n-by-m matrix whose (i,j)th entry is $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_{\omega_j})$. Assuming that $(\boldsymbol{H}_n^t\boldsymbol{H}_n)^{-1}$ exists,

$$\hat{\boldsymbol{\alpha}}_n = \left(\boldsymbol{H}_n^t\boldsymbol{H}_n\right)^{-1}\boldsymbol{H}_n^t\boldsymbol{d}_n. \tag{23}$$

A possible way trade convergence speed for part of the computational complexity involved in determining the least-squares solution (23) has been proposed in [42]. The algorithm, termed Affine Projection algorithm, determines a projection of the solution vector $\boldsymbol{\alpha}$ that solves an under-determined least-squares problem. At each time step n, only the p most recent inputs $\{\boldsymbol{u}_n, \ldots, \boldsymbol{u}_{n-p+1}\}$ and observations $\{d_n, \ldots, d_{n-p+1}\}$ are used. An adaptive algorithm based on this method is derived next.

A. The Kernel Affine Projection Algorithm

In the following, $\boldsymbol{H}_n$ denotes the matrix whose (i,j)th entry is $\kappa(\boldsymbol{u}_{n-i+1}, \boldsymbol{u}_{\omega_j})$, and $\boldsymbol{d}_n$ is the column vector whose ith element is d_{n-i+1}. Our approach starts with the affine projection problem at time step n

$$\min_{\boldsymbol{\alpha}} \|\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}_{n-1}\|^2 \quad \text{subject to} \quad \boldsymbol{d}_n = \boldsymbol{H}_n\boldsymbol{\alpha}. \tag{24}$$

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

$$\mathcal{A}_i = \left\{\boldsymbol{\alpha} : \boldsymbol{h}_{n-i+1}^t\boldsymbol{\alpha} - d_{n-i+1} = 0\right\}, \quad i = 1, \ldots, p$$

with $\boldsymbol{h}_{n-i+1} = [\kappa(\boldsymbol{u}_{n-i+1}, \boldsymbol{u}_{\omega_1})\kappa(\boldsymbol{u}_{n-i+1}, \boldsymbol{u}_{\omega_2})\ldots]^t$. At iteration n, upon the arrival of new data, one of the following alternatives holds. If $\kappa(\cdot, \boldsymbol{u}_n)$ does not satisfy the coherence-based sparsification rule (9), the dictionary remains unaltered. On the other hand, if (9) is met, $\kappa(\cdot, \boldsymbol{u}_n)$ is inserted into the dictionary where it is denoted by $\kappa(\cdot, \boldsymbol{u}_{\omega_{m+1}})$. The number of columns of matrix $\boldsymbol{H}_n$ then is increased by one, relative to $\boldsymbol{H}_{n-1}$, by appending $[\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_{m+1}}) \ldots \kappa(\boldsymbol{u}_{n-p+1}, \boldsymbol{u}_{\omega_{m+1}})]^t$. One more entry is also added to the vector $\hat{\boldsymbol{\alpha}}_n$.

B. First Case Study: $\max_{j=1,\ldots,m} |\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_j})| > \mu_0$

In this case $\kappa(\cdot, \boldsymbol{u}_n)$ can be reasonably well represented by the kernel functions already in the dictionary. Thus, it does not need to be inserted into the dictionary. The solution to (24) can be determined by minimizing the Lagrangian function

$$J(\boldsymbol{\alpha}, \boldsymbol{\lambda}) = \|\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}_{n-1}\|^2 + \boldsymbol{\lambda}^t(\boldsymbol{d}_n - \boldsymbol{H}_n\boldsymbol{\alpha}) \tag{25}$$

where $\boldsymbol{\lambda}$ is the vector of Lagrange multipliers. Differentiating this expression with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\lambda}$, and setting the derivatives to zero, we get the following equations that $\hat{\boldsymbol{\alpha}}_n$ must satisfy

$$2(\hat{\boldsymbol{\alpha}}_n - \hat{\boldsymbol{\alpha}}_{n-1}) = \boldsymbol{H}_n^t\boldsymbol{\lambda} \tag{26}$$

$$\boldsymbol{H}_n\hat{\boldsymbol{\alpha}}_n = \boldsymbol{d}_n. \tag{27}$$

TABLE I
THE KAP ALGORITHM WITH COHERENCE CRITERION

```
Initialization
        Fix the memory length p, the step-size η, and the regularization factor ε
        Insert κ(·, u_p) into the dictionary, denote it by κ(·, u_{ω_1})
        H_p = [κ(u_p, u_{ω_1}) ... κ(u_1, u_{ω_1})]^t,  α̂_p = 0,  m = 1
At each time step n > p, repeat
    1. Get (u_n, d_n)
    2. If max_{j=1,...,m} |κ(u_n, u_{ω_j})| > μ_0: (parameter update)
        Compute the p-by-m matrix H_n = [κ(u_{n-i+1}, u_{ω_j})]_{i=1,...,p; j=1,...,m}
        Calculate α̂_n using equation (28)
    3. If max_{j=1,...,m} |κ(u_n, u_{ω_j})| ≤ μ_0: (parameter update with order increase)
        m = m + 1
        Insert κ(·, u_n) into the dictionary, denote it by κ(·, u_{ω_m})
        Compute the p-by-m matrix H_n = [κ(u_{n-i+1}, u_{ω_j})]_{i=1,...,p; j=1,...,m}
        Calculate α̂_n using equation (30)
```

Assuming $\boldsymbol{H}_n\boldsymbol{H}_n^t$ nonsingular, these equations lead to $\boldsymbol{\lambda} = 2(\boldsymbol{H}_n\boldsymbol{H}_n^t)^{-1}(\boldsymbol{d}_n - \boldsymbol{H}_n\hat{\boldsymbol{\alpha}}_{n-1})$. Substituting into (26), we obtain a recursive update equation for $\hat{\boldsymbol{\alpha}}_n$

$$\hat{\boldsymbol{\alpha}}_n = \hat{\boldsymbol{\alpha}}_{n-1} + \eta\boldsymbol{H}_n^t\left(\epsilon\boldsymbol{I} + \boldsymbol{H}_n\boldsymbol{H}_n^t\right)^{-1}(\boldsymbol{d}_n - \boldsymbol{H}_n\hat{\boldsymbol{\alpha}}_{n-1}) \quad (28)$$

where we have introduced the step-size control parameter η, and the regularization factor $\epsilon\boldsymbol{I}$. At each time step n, (28) requires inverting the usually small p-by-p matrix $(\epsilon\boldsymbol{I} + \boldsymbol{H}_n\boldsymbol{H}_n^t)$.

C. Second Case Study: $\max_{j=1,\ldots,m}|\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_j})| \leq \mu_0$

In this case, $\kappa(\cdot, \boldsymbol{u}_n)$ cannot be represented by the kernel functions already in the dictionary. Then, it is inserted into the dictionary and will henceforth be denoted by $\kappa(\cdot, \boldsymbol{u}_{\omega_{m+1}})$. The order m of (22) is increased by one, and $\boldsymbol{H}_n$ is updated to a p-by-$(m+1)$ matrix. To accommodate the new element in $\hat{\boldsymbol{\alpha}}_n$, we modify (24) as

$$\min_{\boldsymbol{\alpha}} ||\boldsymbol{\alpha}_{1,\ldots,m} - \hat{\boldsymbol{\alpha}}_{n-1}||^2 + \alpha_{m+1}^2 \text{ subject to } \boldsymbol{d}_n = \boldsymbol{H}_n\boldsymbol{\alpha} \quad (29)$$

where $\boldsymbol{\alpha}_{1,\ldots,m}$ denotes the first m elements of the vector $\boldsymbol{\alpha}$ and $\boldsymbol{H}_n$ has been increased by one column as explained before. Note that the $(m+1)$th element α_{m+1} is incorporated to the objective function as a regularizing term. Considerations similar to those made to obtain (28) lead to the following recursion:

$$\hat{\boldsymbol{\alpha}}_n = \begin{bmatrix}\hat{\boldsymbol{\alpha}}_{n-1}\\ 0\end{bmatrix} + \eta\boldsymbol{H}_n^t\left(\epsilon\boldsymbol{I} + \boldsymbol{H}_n\boldsymbol{H}_n^t\right)^{-1}\left(\boldsymbol{d}_n - \boldsymbol{H}_n\begin{bmatrix}\hat{\boldsymbol{\alpha}}_{n-1}\\ 0\end{bmatrix}\right). \quad (30)$$

We call the set of recursions (28) and (30) the Kernel Affine Projection (KAP) algorithm. It is described in pseudocode in Table I. The value of p is termed the memory length or the order of the algorithm. Next, we explore the idea of using instantaneous approximations for the gradient vectors.

D. Instantaneous Approximations—The Kernel NLMS Algorithm

Now consider the case $p = 1$. At each time step n, the algorithm described earlier then enforces $d_n = \boldsymbol{h}_n^t\boldsymbol{\alpha}_n$ where $\boldsymbol{h}_n$ is the column vector whose ith entry is $\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_i})$. Relations (28) and (30) reduce to

1) If $\max_{j=1,\ldots,m}|\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_j})| > \mu_0$

$$\hat{\boldsymbol{\alpha}}_n = \hat{\boldsymbol{\alpha}}_{n-1} + \frac{\eta}{\epsilon + ||\boldsymbol{h}_n||^2}\left(d_n - \boldsymbol{h}_n^t\hat{\boldsymbol{\alpha}}_{n-1}\right)\boldsymbol{h}_n, \quad (31)$$

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

2) If $\max_{j=1,\ldots,m}|\kappa(\boldsymbol{u}_n, \boldsymbol{u}_{\omega_j})| \leq \mu_0$

$$\hat{\boldsymbol{\alpha}}_n = \begin{bmatrix}\hat{\boldsymbol{\alpha}}_{n-1}\\ 0\end{bmatrix} + \frac{\eta}{\epsilon + ||\boldsymbol{h}_n||^2}\left(d_n - \boldsymbol{h}_n^t\begin{bmatrix}\hat{\boldsymbol{\alpha}}_{n-1}\\ 0\end{bmatrix}\right)\boldsymbol{h}_n \quad (32)$$

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

The form of these recursions is that of the normalized LMS algorithm with kernels, referred to as KNLMS and described in pseudocode in Table II. As opposed to the scalar-valued *a priori* error, $e_{a,n} = d_n - \boldsymbol{h}_n^t\hat{\boldsymbol{\alpha}}_{n-1}$, used by KNLMS, we note that KAP algorithm uses a vector-valued error $\boldsymbol{e}_{a,n} = \boldsymbol{d}_n - \boldsymbol{H}_n\hat{\boldsymbol{\alpha}}_{n-1}$, to update the weight vector estimate. The next subsection discusses computational requirements of both approaches.

E. Computational Complexity

Table III reports the estimated computational costs of KAP and KNLMS algorithms for real-valued data, in terms of the number of real multiplications and real additions per iteration. The computation cost to evaluate $\boldsymbol{h}_n$ scales linearly with the dictionary dimension m. This cost has not been included in Table III because it depends on the selected kernel. Recursions with [see (30) and (32)] and without [see (28) and (31)] order increase are considered separately in Table III. The coherence criterion (9) used to select which update to perform is significantly simpler than the approximate linear condition (6) since it consists of comparing the largest element in magnitude of $\boldsymbol{h}_n$ to a threshold μ_0. Note that the final size of a dictionary of kernel functions determined under the rule (9) is finite. This implies that, after a transient period during which the order of the model increases, computational complexity is reduced to that of (28) and (31). The main conclusion is that the costs of KNLMS and KAP algorithms are $\mathcal{O}(m)$ and $\mathcal{O}(p^2m)$, respectively. As illustrated in the next section, the size m of kernel expansions never exceeded a few tens.

TABLE II
THE KNLMS ALGORITHM WITH COHERENCE CRITERION

```
Initialization
        Fix the step-size η, and the regularization factor ε
        Insert κ(·, u_1) into the dictionary, denote it by κ(·, u_{ω_1})
        h_1 = κ(u_1, u_{ω_1}),  α̂_1 = 0,  m = 1
At each time step n > 1, repeat
      1. Get (u_n, d_n)
      2. If max_{j=1,...,m} |κ(u_n, u_{ω_j})| > μ_0: (parameter update)
           Compute the column vector h_n = [κ(u_n, u_{ω_1}) ... κ(u_n, u_{ω_m})]^t
           Update α̂_n using equation (31)
      3. If max_{j=1,...,m} |κ(u_n, u_{ω_j})| ≤ μ_0: (parameter update with order increase)
           m = m + 1
           Insert κ(·, u_n) into the dictionary, denote it by κ(·, u_{ω_m})
           Compute the column vector h_n = [κ(u_n, u_{ω_1}) ... κ(u_n, u_{ω_m})]^t
           Update α̂_n using equation (32)
```

V. SIMULATION EXAMPLES

The purpose of this section is to illustrate the performance of the proposed approach. We shall report the results of two simulated data experiments.

A. Experiment With KNLMS

As a first benchmark problem, we consider the nonlinear system described by the difference equation

$$d_n = \left(0.8 - 0.5\exp\left(-d_{n-1}^2\right)\right) d_{n-1} - \left(0.3 + 0.9\exp\left(-d_{n-1}^2\right)\right) d_{n-2} + 0.1\sin(d_{n-1}\pi) \quad (33)$$

where d_n is the desired output. This highly nonlinear time series has been investigated in [18]. The data were generated by iterating the above equation from the initial condition (0.1,0.1). Outputs d_n were corrupted by a measurement noise sampled from a zero-mean Gaussian distribution with standard deviation equal to 0.1. This led to a signal-to-noise ratio (SNR), defined as the ratio of the powers of d_n and the additive noise, of 17.2 dB. These data were used to estimate a nonlinear model of the form $d_n = \psi(d_{n-1}, d_{n-2})$. In identifying the system, we restricted ourselves to KNLMS and the experimental setup described in [18]. In particular, as in [18], the Gaussian kernel $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp(-3.73\|\boldsymbol{u}_i - \boldsymbol{u}_j\|^2)$ was considered. Preliminary experiments were conducted as explained below to determine all the adjustable parameters, that is, the threshold μ_0, the step-size η and the regularization factor ϵ. The algorithm was then evaluated on several independent test signals, which led to the learning curve depicted in Fig. 1 and the normalized mean-square prediction error reported in Table IV. The same procedure was followed to parameterize and test the state-of-the-art methods discussed later.

The preliminary experiments were conducted on sequences of 3000 samples to determine μ_0, η, and ϵ. Performance was measured in steady state using the mean-square prediction error $\sum_{n=2501}^{3000}(d_n - \psi_{n-1}(\boldsymbol{u}_n))^2$ over the last 500 samples of each sequence, and averaged over 10 independent trials. The dictionary was initialized with $\kappa(\cdot, \boldsymbol{u}_1)$, where $\boldsymbol{u}_1 = [0.1, 0.1]^t$. The step-size η and the regularization coefficient ϵ were determined by grid search over $(10^{-4} \le \eta \le 10^{-1}) \times (10^{-4} \le \epsilon \le 10^{-1})$ with increment 2×10^{-k} within each range $[10^{-k}, 10^{-k+1}]$. The threshold μ_0 was varied from 0.05 to 0.95 in increments of 0.05. It was observed that increasing μ_0 was associated with performance improvements until a threshold was attained, when performance stayed basically unchanged. A practical compromise between the model order and its performance was reached by setting the threshold μ_0 to 0.5. The step-size parameter η and the regularization coefficient ϵ were fixed to 9×10^{-2} and 3×10^{-2}, respectively.

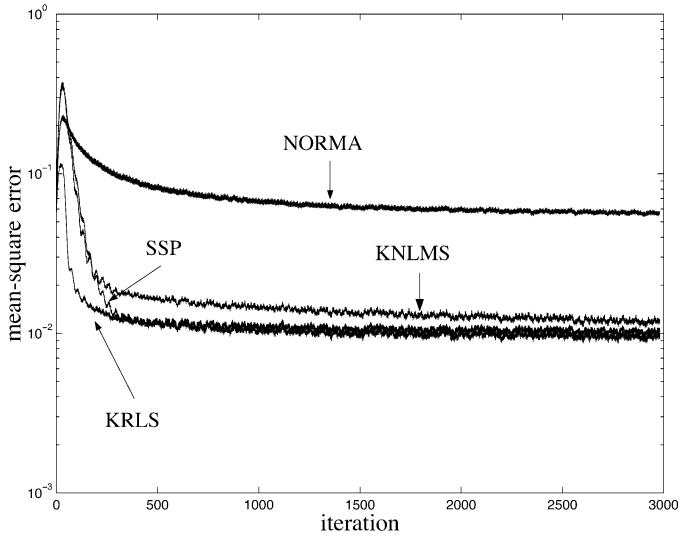

Fig. 1. Learning curves for KNLMS, NORMA, SSP, and KRLS obtained by averaging over 200 experiments.

TABLE III
ESTIMATED COMPUTATIONAL COST PER ITERATION OF KNLMS AND KAP ALGORITHMS

		KNLMS	KAP
without order increase	×	$3m+1$	$(p^2+2p)m + p^3 + p$
	+	$3m$	$(p^2+2p)m + p^3 + p^2$
with order increase	×	$3m+3$	$(p^2+2p)m + p^3 + 2p^2 + p$
	+	$3m+1$	$(p^2+2p)m + p^3 + p^2 + p - 1$

The KNLMS algorithm was tested with the parameter settings specified above over two hundred 10 000-sample independent sequences. This led to the ensemble-average learning curve shown in Fig. 1. The order m of kernel expansions was, on average, equal to 21.3. The normalized mean-square prediction error over the last 5000 samples was determined from

$$\text{NMSE} = E\left\{\frac{\sum_{n=5001}^{10000}\left(d_n - \psi_{n-1}(\boldsymbol{u}_n)\right)^2}{\sum_{n=5001}^{10000} d_n^2}\right\} \quad (34)$$

TABLE IV
EXPERIMENT A: ESTIMATED COMPUTATIONAL COST PER ITERATION, EXPERIMENTAL SETUP, AND PERFORMANCE ON INDEPENDENT TEST SEQUENCES

Algorithm	$\times$	$+$	Parameter settings	m	NMSE
NORMA	$2m$	m	$\lambda = 0.98$, $\eta_n = 1/\sqrt{n}$	38	0.1051
KNLMS	$3m+1$	$3m$	$\mu_0 = 0.5$, $\epsilon = 0.03$, $\eta = 0.09$	21.3	0.0197
SSP	$3m^2+6m+1$	$3m^2+m-1$	$\kappa = 0.001$, $\eta = 0.1$	23.8	0.0184
KRLS	$4m^2+4m$	$4m^2+4m+1$	$\nu = 0.6$	22.1	0.0173

TABLE V
EXPERIMENT B: ESTIMATED COMPUTATIONAL COST PER ITERATION, EXPERIMENTAL SETUP, AND PERFORMANCE ON INDEPENDENT TEST SEQUENCES

Algorithm	$\times$	$+$	Parameter settings	m	NMSE
NORMA	$2m$	m	$\lambda = 0.09$, $\eta_n = 0.09/\sqrt{n}$	35	0.56
KNLMS	$3m+1$	$3m$	$\mu_0 = 0.3$, $\epsilon = 0.0009$, $\eta = 0.01$	5.4	0.20
$\text{KAP}_{p=2}$	$8m+10$	$8m+12$	$\mu_0 = 0.3$, $\epsilon = 0.07$, $\eta = 0.009$	5.4	0.21
$\text{KAP}_{p=3}$	$15m+30$	$15m+36$	$\mu_0 = 0.3$, $\epsilon = 0.07$, $\eta = 0.01$	5.4	0.21
SSP	$3m^2+6m+1$	$3m^2+m-1$	$\kappa = 0.005$, $\eta = 0.03$	17.5	0.21
KRLS	$4m^2+4m$	$4m^2+4m+1$	$\nu = 0.7$	8.1	0.17

where the expectation was approximated by averaging over the ensemble. As reported in Table IV, the NMSE was found to be 0.0197. For comparison purposes, state-of-the-art kernel-based methods for online prediction of time series were also considered: NORMA [43], sparse sequential projection (SSP) [18], and KRLS [20].

As the KNLMS algorithm, NORMA performs stochastic gradient descent on RKHS. The order of the kernel expansion is fixed *a priori* since it uses the m most recent kernel functions as a dictionary. NORMA requires $\mathcal{O}(m)$ operations per iteration. The SSP algorithm also starts with stochastic gradient descent to calculate the *a posteriori* estimate. The resulting $(m+1)$-order kernel expansion is then projected onto the subspace spanned by the m kernel functions of the dictionary, and the projection error is compared to a threshold in order to evaluate whether the contribution of the $(m+1)$th candidate kernel function is significant enough. If not, the projection is used as the *a posteriori* estimate. In the spirit of the sparsification rule (6), this test requires $\mathcal{O}(m^2)$ operations per iteration when implemented recursively. KRLS is a RLS-type algorithm with an order-update process controlled by (6). Its computational complexity is also $\mathcal{O}(m^2)$ operations per iteration. Table IV reports a comparison of the estimated computational costs per iteration for each algorithm, in the most usual case where no order increase is performed. These results are expressed for real-valued data in terms of the number of real multiplications and real additions. The same procedure used for KNLMS was followed to initialize and test NORMA, SSP, and KRLS. This means that preliminary experiments were conducted on 10 independent 3000-sample sequences to perform explicit grid search over parameter spaces and, following the notations used in [18], [20], and [43], to select the best settings reported in Table IV. Each approach was tested over two hundred 10 000-sample independent sequences, which led to the average orders m and normalized mean-square prediction errors also displayed in this table. As shown in Fig. 1, the algorithms with quadratic complexity performed better than the other two, with only a small advantage of SSP over KNLMS that must be balanced with the large increase in computational cost. This experiment also highlights that KNLMS significantly outperformed NORMA, which demonstrates a clear advantage of the coherence-based sparsification rule.

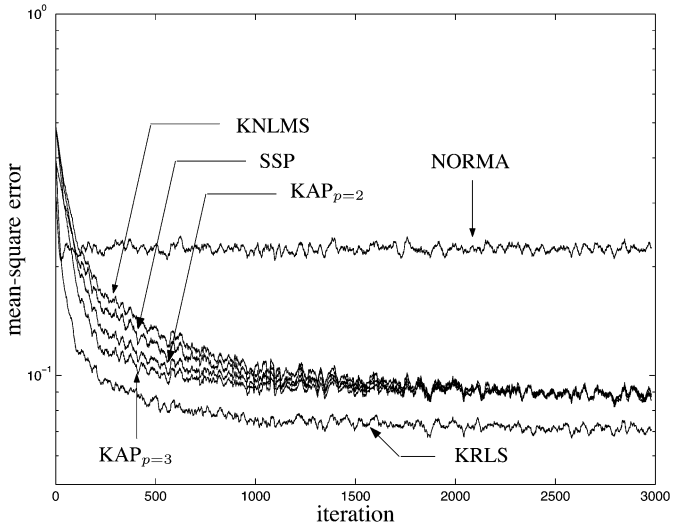

Fig. 2. Learning curves for KAP, KNLMS, SSP, NORMA, and KRLS obtained by averaging over 200 experiments.

B. Experiment With the KAP Algorithm

As a second application, we consider the discrete-time nonlinear dynamical system

$$\begin{cases} v_n = 1.1\exp\left(-|v_{n-1}|\right) + u_n \\ d_n = v_n^2 \end{cases} \tag{35}$$

where u_n and d_n are the input and the desired output, respectively. The data were generated from the initial condition $v_0 = 0.5$. The input u_n was sampled from a zero-mean Gaussian distribution with standard deviation 0.25. The system output d_n was corrupted by an additive zero-mean white Gaussian noise with standard deviation equal to 1, corresponding to a SNR of -4.0 dB. The KAP algorithm was used to identify a model of the form $d_n = \psi(u_n)$. Preliminary experiments were conducted to determine the kernel and, as before, all the adjustable parameters. The algorithm was next evaluated on several independent test signals, which led to the learning curves depicted in Fig. 2 and the normalized mean-square prediction errors reported in Table V.

The preliminary experiments were conducted on sequences of 3000 samples to select the kernel, and determine the best set-

tings for the algorithm. Performance was measured using the mean-square prediction error over the last 500 samples of each sequence, and averaged over 10 independent trials. The dictionary was initialized with $\kappa(\cdot, \boldsymbol{u}_1)$. Three of the most commonly used kernels were considered: the polynomial kernel, the Gaussian kernel and the Laplacian kernel. The latter, defined as $\kappa(\boldsymbol{u}_i, \boldsymbol{u}_j) = \exp(-\|\boldsymbol{u}_i - \boldsymbol{u}_j\|/\beta_0)$, was shown to be the most accurate in this experiment. The bandwidth β_0 was varied from 0.1 to 1 in increments of 0.005 to find the optimal setting. The coherence threshold μ_0 was also varied from 0.05 to 0.5 in increments of 0.05. Memory lengths p ranging from 1 to 3 were considered and, in each case, the best performing step-size parameter η and regularization constant ϵ were determined by grid search over $(10^{-4} \leq \eta \leq 10^{-1}) \times (10^{-4} \leq \epsilon \leq 10^{-1})$ with increment 2×10^{-k} within each range $[10^{-k}, 10^{-k+1}]$. Parameter choices are reported in Table V, for p ranging from 1 to 3.

Each configuration was run over 200 10 000-sample independent test sequences. The order m of the kernel expansion was 5.4 on average, and the mean value of the Babel function was 0.56. By Proposition 1, this indicates that the kernel functions of the dictionary were most frequently, if not always, chosen linearly independent. Steady-state performance was measured by the normalized mean-square prediction error (34). Table V reports mean values over the 200 test sequences for memory lengths p ranging from 1 to 3. It indicates that steady-state performance remained almost unchanged as p increased. Fig. 2 illustrates the convergence behavior of KAP-type methods. These ensemble-average learning curves were obtained by time averaging over 20 consecutive samples. It appears as an evidence that KAP algorithm provided a significant improvement in convergence rate over KNLMS.

The same procedure as before was followed to initialize and test NORMA, SSP and KRLS algorithms. The preliminary experiments that were conducted led to the parameter settings displayed in Table V, where we use the same notations as those in [18], [20], and [43]. This table also reports the average order m of kernel expansions and the normalized mean-square prediction error of each algorithm, estimated over 200 independent test sequences. Fig. 2 shows that KRLS converges faster than KAP-type algorithms, as might be expected, since they are derived from stochastic-gradient approximations. Nevertheless, the KRLS algorithm is an order of magnitude in m costlier than KAP. It can also be seen that SSP has approximately the same convergence rate as KNLMS, but converges slower than the other two KAP algorithms. Moreover, SSP is more demanding computationally and requires kernel expansions of larger order m. Fig. 2 finally highlights that NORMA, the other approach with linear complexity in m, is clearly outperformed by KAP-type algorithms.

The tradeoffs involved in using RLS, affine projection and LMS algorithms are well known in linear adaptive filtering. It is expected that these tradeoffs would persist with their kernel-based counterparts. This was confirmed by simulations, even considering that no theoretical effort was made to determine analytically the optimum tuning parameters for each algorithm. In general, the KRLS algorithm will provide the fastest convergence rate at the expense of the highest computational complexity. The KNLMS algorithm will lead to the lowest computational cost, but will affect the convergence rate of the filtering process. The KAP algorithm lies halfway between these two extremes, converging faster than KNLMS and slower than KRLS, and having a computational complexity that is higher than KNLMS and lower than KRLS.

VI. Conclusion

Over the last 10 years or so there has been an explosion of activity in the field of learning algorithms utilizing reproducing kernels, most notably in the field of classification and regression. The use of kernels is an attractive computational shortcut to create nonlinear versions of conventional linear algorithms. In this paper, we have demonstrated the versatility and utility of this family of methods to develop nonlinear adaptive algorithms for time series prediction, specifically of the KAP and KNLMS types. A common characteristic in kernel-based methods is that they deal with models whose order equals the size of the training set, making them unsuitable for online applications. Therefore, it was essential to first develop a methodology of controlling the increase in the model order as new input data become available. This led us to consider the coherence parameter, a fundamental quantity that characterizes the behavior of dictionaries in sparse approximation problems. The motivation for using it was twofold. First, it offers several attractive properties that can be exploited to assess the novelty of input data. This framework is a core contribution to our paper. Second, the coherence parameter is easy to calculate and its computational complexity is only linear in the dictionary size. We proposed to incorporate it into a kernel-based affine projection algorithm with order-update mechanism, which has also been a notable contribution to our study. Perspectives include the use of the Babel function instead of the coherence parameter since it provides a more in-depth description of a dictionary. Online minimization of the coherence parameter or the Babel function of the dictionary by adding or removing kernel functions also seems interesting. Finally, in a broader perspective, improving our approach with tools derived from compressed sensing appears as a very promising subject of research.

References

[1] G. B. Giannakis and E. Serpedin, "A bibliography on nonlinear system identification," *Signal Process.*, vol. 81, pp. 553–580, 2001.

[2] 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.

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

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

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

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

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

[8] J. Sjöberg, Q. Zhang, L. Ljung, A. Benveniste, B. Deylon, P.-Y. Glorennec, H. Hjalmarsson, and A. Juditsky, "Nonlinear black-box modeling in system identification: A unified overview," *Automatica*, vol. 31, no. 12, pp. 1691–1724, 1995.

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

[10] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, 1950.

[11] M. A. Aizerman, E. M. Braverman, and L. I. Rozonoer, "The method of potential functions for the problem of restoring the characteristic of a function converter from randomly observed points," *Autom. Remote Control*, vol. 25, no. 12, pp. 1546–1556, 1964.

[12] G. Kimeldorf and G. Wahba, "Some results on Tchebycheffian spline functions," *J. Math. Anal. Appl.*, vol. 33, pp. 82–95, 1971.

[13] G. Wahba, *Spline Models for Observational Data*. Philadelphia, PA: SIAM, 1990.
[14] D. L. Duttweiler and T. Kailath, "An RKHS approach to detection and estimation theory: Some parameter estimation problems (Part V)," *IEEE Trans. Inf. Theory*, vol. 19, no. 1, pp. 29–37, 1973.
[15] B. Schölkopf, J. C. Burges, and A. J. Smola, *Advances in Kernel Methods*. Cambridge, MA: MIT Press, 1999.
[16] A. J. Smola and B. Schölkopf, A Tutorial on Support Vector Regression NeuroCOLT, Royal Holloway College, Univ. London, UK, Tech. Rep. NC-TR-98-030, 1998.
[17] J. A. K. Suykens, T. van Gestel, J. de Brabanter, B. de Moor, and J. Vandewalle, *Least Squares Support Vector Machines*. Singapore: World Scientific, 2002.
[18] T. J. Dodd, V. Kadirkamanathan, and R. F. Harrison, "Function estimation in Hilbert space using sequential projections," in *Proc. IFAC Conf. Intell. Control Syst. Signal Process.*, 2003, pp. 113–118.
[19] T. J. Dodd, B. Mitchinson, and R. F. Harrison, "Sparse stochastic gradient descent learning in kernel models," in *Proc. 2nd Int. Conf. Computat. Intell., Robot. Autonomous Syst.*, 2003.
[20] Y. Engel, S. Mannor, and R. Meir, "Kernel recursive least squares," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2275–2285, 2004.
[21] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer, 1995.
[22] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," *Philos. Trans. Roy. Soc. London Ser. A*, vol. 209, pp. 415–446, 1909.
[23] T. J. Dodd and R. F. Harrison, "Estimating Volterra filters in Hilbert space," in *Proc. IFAC Conf. Intell. Control Syst. Signal Process.*, 2003, pp. 538–543.
[24] Y. Wan, C. X. Wong, T. J. Dodd, and R. F. Harrison, "Application of a kernel method in modeling friction dynamics," in *Proc. IFAC World Congress*, 2005.
[25] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K. R. Müller, *Fisher discriminant analysis with kernels* (in Proc. Advances in Neural Networks for Signal Processing), Y. H. Hu, J. Larsen, E. Wilson, and S. Douglas, Eds. San Mateo, CA: Morgan Kaufmann, 1999, pp. 41–48.
[26] F. Abdallah, C. Richard, and R. Lengellé, "An improved training algorithm for nonlinear kernel discriminants," *IEEE Trans. Signal Process.*, vol. 52, no. 10, pp. 2798–2806, 2004.
[27] B. Schölkopf, A. J. Smola, and K. R. Müller, "Nonlinear component analysis as a kernel eigenvalue problem," *Neural Computat.* , vol. 10, no. 5, pp. 1299–1319, 1998.
[28] B. Schölkopf, R. Herbrich, and R. Williamson, A Generalized Representer Theorem NeuroCOLT, Royal Holloway College, Univ. London, UK, Tech. Rep. NC2-TR-2000-81, 2000.
[29] L. Hoegaerts, "Eigenspace methods and subset selection in kernel based learning," Ph.D. thesis, Katholieke Univ. Leuven, Leuven, Belgium, 2005.
[30] B. J. de Kruif and T. J. A. de Vries, "Pruning error minimization in least squares support vector machines," *IEEE Trans. Neural Netw.*, vol. 14, no. 3, pp. 696–702, 2003.
[31] J. A. K. Suykens, J. de Brabanter, L. Lukas, and J. Vandewalle, "Weighted least squares support vector machines: Robustness and sparse approximation," *Neurocomput.*, vol. 48, pp. 85–105, 2002.
[32] G. C. Cawley and N. L. C. Talbot, "Improved sparse least-squares support vector machines," *Neurocomput.*, vol. 48, pp. 1025–1031, 2002.
[33] L. Hoegaerts, J. A. K. Suykens, J. Vandewalle, and B. de Moor, "Subset based least squares subspace regression in RKHS," *Neurocomput.*, vol. 63, pp. 293–323, 2005.
[34] B. Schölkopf, S. Mika, C. J. C. Burges, P. Knirsch, K. R. Müller, G. Rätsch, and A. J. Smola, "Input space versus feature space in kernel-based methods," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1000–1017, 1999.
[35] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. Inf. Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.
[36] S. Mallat and Z. Zhang, "Matching pursuit with time-frequency dictionaries," *IEEE Trans. Signal Process.*, vol. 41, no. 12, pp. 3397–3415, 1993.
[37] D. L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Trans. Inf. Theory*, vol. 47, no. 7, pp. 2845–2862, 2001.
[38] M. Elad and A. M. Bruckstein, "A generalized uncertainty principle and sparse representations in pairs of bases," *IEEE Trans. Inf. Theory*, vol. 48, no. 9, pp. 2558–2567, 2002.
[39] D. L. Donoho and M. Elad, "Optimally sparse representation in general (nonorthogonal) dictionaries via ℓ_1 minimization," in *Proc. Nat. Acad. Sci. USA*, 2003, vol. 100, pp. 2197–2202.
[40] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge, U.K.: Cambridge Univ. Press, 1985.
[41] M. Girolami, "Orthogonal series density estimation and the kernel eigenvalue problem," *Neural Computat.*, vol. 14, pp. 669–688, 2002.
[42] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Electron. Commun. Japan*, vol. 67-A, pp. 19–27, 1984.
[43] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, 2004.

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 Compiègne University of Technology, France, all in electrical and computer engineering.

From 1999 to 2003, he was an Associate Professor with Troyes University of Technology, Troyes, France. Since 2003, he has been a Professor with the Systems Modeling and Dependability Laboratory, Troyes University of Technology. He is also the current director of this laboratory. His research interests include statistical signal processing and machine learning. He is the author of more than 80 papers. In 2005, he was offered the position of chairman of the Ph.D. students network of the federative CNRS research group ISIS on Information, Signal, Images and Vision.

Dr. Richard was the General Chair of the 21th Francophone Conference GRETSI on Signal and Image Processing, held in Troyes, in 2007. He is a member of the GRETSI Association Board. He was recently nominated EURASIP liaison local officer for France. He serves also as an Associate Editor of the IEEE TRANSACTIONS ON SIGNAL PROCESSING and of the *Research Letters in Signal Processing*. He is currently a member of the Signal Processing Theory and Methods Technical Committee of the IEEE Signal Processing Society.

José Carlos M. Bermudez (S'78–M'85–SM'02) received the B.E.E. degree from Federal University of Rio de Janeiro (UFRJ), Rio de Janeiro, Brazil, the M.Sc. degree in electrical engineering from COPPE/UFRJ, and the Ph.D. degree in electrical engineering from Concordia University, Montreal, Canada, in 1978, 1981, and 1985, respectively.

He joined the Department of Electrical Engineering, Federal University of Santa Catarina (UFSC), Florianópolis, Brazil, in 1985, where he is currently a Professor of electrical engineering. In winter 1992, he was a Visiting Researcher with the Department of Electrical Engineering, Concordia University. In 1994, he was a Visiting Researcher with the Department of Electrical Engineering and Computer Science, University of California, Irvine. His research interests have involved analog signal processing using continuous-time and sampled-data systems. His recent research interests are in digital signal processing, including linear and nonlinear adaptive filtering, active noise and vibration control, echo cancellation, image processing, and speech processing.

Prof. Bermudez served as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING in the area of adaptive filtering from 1994 to 1996 and from 1999 to 2001, and as the Signal Processing Associate Editor for the *Journal of the Brazilian Telecommunications Society* (2005–2006). He was a member of the Signal Processing Theory and Methods Technical Committee of the IEEE Signal Processing Society from 1998 to 2004. He is currently an Associate Editor for the *EURASIP Journal on Advances in Signal Processing*.

Paul Honeine (M'07) was born in Beirut, Lebanon, on October 2, 1977. He received the Dipl.-Ing. degree in mechanical engineering in 2002 and the M.Sc. degree in industrial control in 2003, both from the Faculty of Engineering, the Lebanese University, Lebanon. In 2007, he received the Ph.D. degree in system optimization and security from the University of Technology of Troyes, France.

He was a Postdoctoral Research Associate with the Systems Modeling and Dependability Laboratory, University of Technology of Troyes, from 2007 to 2008. Since September 2008, he has been an Assistant Professor with the University of Technology of Troyes. His research interests include nonstationary signal analysis, nonlinear adaptive filtering, sparse representations, machine learning, and wireless sensor networks.