The Generalized Complex Kernel Least-Mean-Square Algorithm

Rafael Boloix-Tortosa*, *Member, IEEE,* Juan José Murillo-Fuentes, *Senior Member, IEEE,* Sotirios A. Tsaftaris, *Senior Member, IEEE*

***Abstract*—We propose a novel adaptive kernel-based regression method for complex-valued signals: the generalized complex-valued kernel least-mean-square (gCKLMS). We borrow from the new results on widely linear reproducing kernel Hilbert space (WL-RKHS) for nonlinear regression and complex-valued signals, recently proposed by the authors. This paper shows that in the adaptive version of the kernel regression for complex-valued signals we need to include another kernel term, the so-called pseudo-kernel. This new solution is endowed with better representation capabilities in complex-valued fields since it can efficiently decouple the learning of the real and the imaginary part. Also, we review previous realizations of the complex KLMS algorithm and its augmented version to prove that they can be rewritten as particular cases of the gCKLMS. Furthermore, important conclusions on the design of the kernels are drawn that help to greatly improve the convergence of the algorithms. In the experiments, we revisit the nonlinear channel equalization problem to highlight the better convergence of the gCKLMS compared to previous solutions. Also, the flexibility of the proposed generalized approach is tested in a second experiment with non-independent real and imaginary parts. The results illustrate the significant performance improvements of the gCKLMS approach when the complex-valued signals have different properties for the real and imaginary parts.**

***Index Terms*—LMS, complex-valued, RKHS, kernel methods.**

I. Introduction

COMPLEX-VALUED signals model many systems in diverse applications such as electromagnetism, telecommunications, optics or acoustics, among others. Complex-valued signal processing is thus of fundamental interest as it provides a natural way to represent some signals and transformations involved in those systems. While the linear case has been widely studied (see for example [1] and references therein), nonlinear processing still remains an open problem. Nonlinear processing of complex-valued signals has been tackled, among others, from the point of view of neural networks [2], [3], nonlinear adaptive filtering [4], or reproducing kernel Hilbert spaces (RKHS) [5]. This latter field is gaining increasing interest within the signal processing community as it provides a simple but elegant way to treat nonlinearities. Complex kernel-based algorithms have been lately proposed for regression [6], [7], [8], kernel principal component analysis [9] or classification [10].

Regarding complex-valued regression within the RKHS framework, we have recently highlighted in [11] the need of a new term: the *pseudo-kernel*. We redefined the kernel-based regularized least squares regression to include the pseudo-kernel, and the resulting structure resembles that of the widely linear (WL) solutions, being capable of learning any complex-valued function effectively. As discussed in [11], the need for a pseudo-kernel can be justified in cases where the real and imaginary parts are correlated and learning them independently is, at best, suboptimal. Also, a pseudo-kernel is needed when the real and imaginary parts are not best represented by the same kernel, i.e., the same measure of similarity. Furthermore, we analyzed in [11] the structure of the kernel and pseudo-kernel, and discussed how to design these functions, and when should they be real or complex-valued. As a result, two important remarks were made. First, if the real and imaginary parts of the output are independent, then the kernel and pseudo-kernel should be real-valued. Second, if the real and imaginary parts of the output have different properties in terms of similarity, the pseudo-kernel is needed. On the contrary, the pseudo-kernel vanishes if the real and imaginary parts of the output are independent but have the same properties in terms of similarity, i.e., the same kernel can be used for the real and imaginary parts.

The complex kernel least-mean-square (CKLMS) allows for an adaptive version of the RKHS techniques. In the literature, there are several proposals for CKLMS algorithms. The authors in [6] address the problem of adaptive filtering of complex signals and calculate the gradient of cost functions by using Wirtinger's derivatives. Two alternatives are described. The first alternative, denoted by CKLMS1, uses real kernels developed by means of the *complexification technique* of real RKHSs. The second one, the CKLMS2, proposes the use of complex kernels, in particular, the complex Gaussian kernel [10]. With these two alternatives, they develop two realizations of the kernel least-mean-square (KLMS) algorithm [12] for complex signals. The same complex Gaussian kernel is also adopted in [13] and in [14]. Augmented or WL filters consider both the original values of the signal data and their conjugates [15]. In [14], the authors introduce WL adaptive filters to develop an augmented version of the complex KLMS algorithm, the ACKLMS. In the light of the theoretical framework in [11], in this paper we review these results on the CKLMS to propose a new approach, as follows.

R. Boloix-Tortosa, and J.J. Murillo-Fuentes are with the Dep. de Teoría de la Señal y Comunicaciones, Escuela Técnica Superior de Ingeniería, Universidad de Sevilla, Camino de los Descubrimientos sn, 41092 Sevilla, Spain. e-mail: {rboloix,murillo}@us.es.

S. A. Tsaftaris is with the School of Engineering, The University of Edinburgh, Edinburgh EH8 3FB, U.K., and also with The Alan Turing Institute, London NW1 2DB, U.K.

Thanks to the Spanish Government (Ministerio de Economía y Competitividad, TEC2016-78434-C3-02-R, and Ministerio de Educación, Cultura y Deporte, Subprograma Estatal de Movilidad (PRX18/00523), del Plan Estatal de I+D+I) and European Union (FEDER) for funding.

Previous augmented or WL versions of the complex KLMS use limited feature mappings. The KLMS algorithm [12] considers a linear input-output mapping but on the transformed inputs by using the feature map, $\hat{\mathbf{y}} = \langle \Phi(\mathbf{x}), \mathbf{w} \rangle$. For the ACKLMS algorithm [14] they propose to use a WL model with two linear terms, one with the feature map transformation and the other with the conjugate of this transformation: $\hat{\mathbf{y}} = \langle \Phi(\mathbf{x}), \mathbf{w} \rangle + \langle \Phi(\mathbf{x})^*, \mathbf{v} \rangle$. We propose a richer way to represent this model leading to improved results on the adaptive regression task for complex-valued signals. This is the representation of a complex-valued function as a two-dimensional real-valued vector function obtained by stacking the real part on top of the imaginary part. From the theory of kernels for learning vector-valued functions [16] we are able to propose a suitable feature map transformation in this composite representation.

By exploiting this composite feature map, we can set a linear input-output mapping on the transformed inputs to derive the KLMS algorithm for the composite representation. This is the second contribution of the paper. In this new composite KLMS algorithm we prove that the kernel should be matrix-valued. Once the algorithm has been developed in composite representation we develop its complex value representation, which is the third and central contribution of this paper. This is an augmented or WL version of the CKLMS. To avoid confusion with previous proposals we denote it by generalized CKLMS (gCKLSM). It has both a kernel and a pseudo-kernel, in accordance with the general theory for complex RKHS in [11]. This representation exhibits better reproducing capabilities than previous approaches [17], [18], [19], [14]. Furthermore, we show that these previous proposals are particular cases of the gCKLMS.

The right design of the kernel and the pseudo-kernel is important to properly model the system. As a final contribution we analyze different options and explain their implications. The kernel and pseudo-kernel in the gCKLSM have the same structures introduced in [11], this is our starting point to enhance the performance of the gCKLMS. These results can also be used to improve previous proposals.

This paper is organized as follows. We devote Section II to review the theory of learning in RKHS of vector-valued functions [16] as a way to find a suitable feature map transformation. In Section III we use the feature map to develop the KLMS algorithm for the composite representation. The formulation for the gCKLMS algorithm is found in Section IV. In this section we also show the equations for the kernel and pseudo-kernel terms. In Section V we compare the gCKLMS with other complex KLMS algorithms in the literature to show that they are particular cases of the gCKLMS. Experiments are included in Section VI, where the gCKLMS algorithm is tested first in the context of a nonlinear channel equalization task, and then in the learning of samples of a filtered random process. These experiments show that the gCKLMS outperforms other KLMS algorithms because it takes advantage of having both the kernel and the pseudo-kernel. We end the paper with some conclusions in Section VII.

In the notation used throughout the paper, bold lower-case letters are used to denote vectors, while matrices are denoted using bold upper-case letters. For matrix $\mathbf{A}$, $[\mathbf{A}]_{l,q}$ is its (l,q) entry. To denote the i-th sample of a vector or signal we use, respectively, $\mathbf{a}(i)$ and $a(i)$. $\mathcal{R}\{a\}$ is the real part of a. Transpose operation is represented by $^\top$, while $^\mathrm{H}$ represents the Hermitian and * complex conjugation. $\mathbb{E}[\cdot]$ is the expectation operator.

II. RKHS OF COMPOSITE VECTOR-VALUED FUNCTIONS

A complex function $f(\mathbf{x}) = f_\mathrm{r}(\mathbf{x}) + \mathrm{j} f_\mathrm{j}(\mathbf{x})$ can be represented as a composite vector-valued function $\mathbf{f}_\mathbb{R}(\mathbf{x}) = [f_\mathrm{r}(\mathbf{x}) \;\; f_\mathrm{j}(\mathbf{x})]^\top \in \mathbb{R}^2$, also known as the dual real channel (DRC) formulation, by stacking its real part on its imaginary part. The definition of the RKHS for vector-valued functions [16] parallels the one for scalar functions [20], with the main difference that the reproducing kernel is now matrix-valued [21], [16].

Let $\mathcal{H}$ be a Hilbert space of functions $\mathbf{f}$ on a set $\mathcal{X}$ with values in $\mathcal{Y}$. $\mathcal{H}$ is a RKHS when for any $\mathbf{x} \in \mathcal{X}$ and any $\mathbf{y} \in \mathcal{Y}$ the linear functional which maps $\mathbf{f}$ to $(\mathbf{y}, \mathbf{f}(\mathbf{x}))_\mathcal{Y}$ is continuous on $\mathcal{H}$ [16]. Here, $(\cdot,\cdot)_\mathcal{Y}$ represents the inner product in the Hilbert space $\mathcal{Y}$, while $\langle \cdot,\cdot \rangle_\mathcal{H}$ is the inner product in $\mathcal{H}$.

From the Riesz Lemma, for every $\mathbf{x} \in \mathcal{X}$ and $\mathbf{y} \in \mathcal{Y}$ there is a linear operator $\mathbf{K}_\mathbf{x} : \mathcal{Y} \to \mathcal{H}$, such that $(\mathbf{y}, \mathbf{f}(\mathbf{x}))_\mathcal{Y} = \langle \mathbf{K}_\mathbf{x}\mathbf{y}, \mathbf{f} \rangle_\mathcal{H}$. Let us now introduce the linear operator $\mathbf{K}(\mathbf{x},\mathbf{x}') : \mathcal{Y} \to \mathcal{Y}$, for every $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$, defined by $\mathbf{K}(\mathbf{x},\mathbf{x}')\mathbf{y} := (\mathbf{K}_{\mathbf{x}'}\mathbf{y})(\mathbf{x})$.

We say that $\mathbf{K} : \mathcal{X} \times \mathcal{X} \to \mathcal{L}(\mathcal{Y})$, where $\mathcal{L}(\mathcal{Y})$ denotes the set of all bounded linear operators from $\mathcal{Y}$ to itself, is a matrix-valued kernel [16] (or operator-valued kernel if $\mathcal{Y}$ is not finite dimensional [22]) if it satisfies the following properties for every $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$:

- (*a*) For every $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$, we have $(\mathbf{y}, \mathbf{K}(\mathbf{x},\mathbf{x}')\mathbf{y}')_\mathcal{Y} = \langle \mathbf{K}_\mathbf{x}\mathbf{y}, \mathbf{K}_{\mathbf{x}'}\mathbf{y}' \rangle_\mathcal{H}$.
- (*b*) $\mathbf{K}(\mathbf{x},\mathbf{x}') = \bar{\mathbf{K}}(\mathbf{x}',\mathbf{x})$, and $\mathbf{K}(\mathbf{x},\mathbf{x}) \in \mathcal{L}_+(\mathcal{Y})$, where $\bar{\mathbf{K}}$ denotes the adjoint and $\mathcal{L}_+(\mathcal{Y})$ the set of all positive semi-definite bounded linear operators, i.e., $(\mathbf{y}, \mathbf{K}(\mathbf{x},\mathbf{x})\mathbf{y})_\mathcal{Y} \geqslant 0$ for any $\mathbf{y} \in \mathcal{Y}$.
- (*c*) For any positive integer m, we have that $\sum_{l,q \in \{1,\cdots,m\}} (\mathbf{y}_q, \mathbf{K}(\mathbf{x}_q,\mathbf{x}_l)\mathbf{y}_l)_\mathcal{Y} \geqslant 0$, for any $\mathbf{x}_l, \mathbf{x}_q \in \mathcal{X}$, $\mathbf{y}_l, \mathbf{y}_q \in \mathcal{Y}$.

Proof of these properties can be found in [16]. Also, it can be shown that if $\mathbf{K}$ is a kernel then there exists a unique (up to an isometry) RKHS of functions from $\mathcal{X}$ to $\mathcal{Y}$ which admits $\mathbf{K}$ as the reproducing kernel.

In the case of $\mathcal{Y} = \mathbb{R}^2$, the kernel function $\mathbf{K}$ takes values as 2×2 matrices and, from property (*a*), the matrix elements can be found as:

$$[\mathbf{K}(\mathbf{x},\mathbf{x}')]_{l,q} = \langle \mathbf{K}_\mathbf{x}\mathbf{e}_l, \mathbf{K}_{\mathbf{x}'}\mathbf{e}_q \rangle_\mathcal{H}, \tag{1}$$

where $\mathbf{e}_l, \mathbf{e}_q$ are the standard coordinate bases in $\mathbb{R}^2$, for $l, q \in \{1, 2\}$.

A. Feature map

We next define a suitable feature map representation for the matrix-valued kernel that will be later useful in deriving the gCKLMS algorithm.

Every kernel $\mathbf{K}$ admits a feature map representation. A feature map is a continuous function $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{L}(\mathcal{Y}, \mathcal{W})$, where $\mathcal{L}(\mathcal{Y}, \mathcal{W})$ denotes all bounded linear operators from $\mathcal{Y}$ into the feature Hilbert space $\mathcal{W}$ [22]. If $\bar{\boldsymbol{\phi}}(\mathbf{x})$ is the adjoint of $\boldsymbol{\phi}(\mathbf{x})$, it is in $\mathcal{L}(\mathcal{W}, \mathcal{Y})$, then

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \bar{\boldsymbol{\phi}}(\mathbf{x})\boldsymbol{\phi}(\mathbf{x}'), \tag{2}$$

for any $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$.

In the case of finite dimensional Hilbert spaces $\mathcal{Y} = \mathbb{R}^2$ and $\mathcal{W} = \mathbb{R}^m$, relative to standard basis of both spaces $\boldsymbol{\phi}(\mathbf{x})$ is a $m \times 2$ matrix. Each entry of this matrix, $[\boldsymbol{\phi}(\mathbf{x})]_{p,q} = \phi_{pq}(\mathbf{x})$ is a scalar-valued continuous function of $\mathbf{x} \in \mathcal{X}$, and each entry of the kernel is

$$[\mathbf{K}(\mathbf{x}, \mathbf{x}')]_{l,q} = \sum_{i \in \{1, \cdots, m\}} \phi_{il}(\mathbf{x})\phi_{iq}(\mathbf{x}'). \tag{3}$$

Note that when $\mathcal{Y} = \mathbb{R}$, then $\boldsymbol{\phi}(\mathbf{x}) \in \mathcal{W}$, but this is not the case here.

III. The composite KLMS algorithm

Consider the training sequence of input-output pairs $\{(\mathbf{x}(1), y(1)), ..., (\mathbf{x}(N), y(N))\}$ where $y(n) \in \mathbb{C}$ and $\mathbf{x}(n) \in \mathbb{C}^d$. The goal is to uncover the underlying complex-valued function $f(\mathbf{x}(i))$ based on these examples, so that to minimize the mean square error $J = \mathbb{E}[|y(i) - f(\mathbf{x}(i))|^2] = \mathbb{E}[|e(i)|^2]$. By using the composite notation, this can be written as

$$\begin{aligned} J &= \mathbb{E}[|e(i)|^2] = \mathbb{E}[(y_{\mathrm{r}}(i) - f_{\mathrm{r}}(\mathbf{x}(i)))^2 + (y_{\mathrm{j}}(i) - f_{\mathrm{j}}(\mathbf{x}(i)))^2] \\ &= \mathbb{E}[(\mathbf{y}_{\mathbb{R}}(i) - \mathbf{f}_{\mathbb{R}}(\mathbf{x}(i)))^{\top} (\mathbf{y}_{\mathbb{R}}(i) - \mathbf{f}_{\mathbb{R}}(\mathbf{x}(i)))], \end{aligned} \tag{4}$$

where $\mathbf{f}_{\mathbb{R}}(\mathbf{x}) = [f_{\mathrm{r}}(\mathbf{x})\ f_{\mathrm{j}}(\mathbf{x})]^{\top}$ and $\mathbf{y}_{\mathbb{R}} = [y_{\mathrm{r}}\ y_{\mathrm{j}}]^{\top}$.

The least-mean-square (LMS) algorithm would consider a linear input-output mapping, i.e., $f(\mathbf{x}(i)) = \mathbf{w}^{\mathrm{H}}\mathbf{x}(i)$, and compute the weight vector $\mathbf{w}$ adaptively using stochastic gradient descent updates [23]. However, instead of a direct linear input-output mapping, the KLMS [12] is performed on the transformed inputs by using the feature map. We propose here to use the composite notations and the theory for RKHS of composite vector-valued functions described in the previous section. Therefore, we use the feature map $\boldsymbol{\phi} : \mathcal{X} \to \mathcal{L}(\mathcal{Y}, \mathcal{W})$ and set $\mathbf{f}_{\mathbb{R}}(\mathbf{x}) = \bar{\boldsymbol{\phi}}(\mathbf{x})\mathbf{w}$, where $\mathbf{w} \in \mathcal{W}$.

Note that in the general case $\mathcal{W}$ could be an infinite dimensional Hilbert space. For the particular case of $\mathcal{W} = \mathbb{R}^m$, since $\mathcal{Y} = \mathbb{R}^2$ then $\boldsymbol{\phi}(\mathbf{x}) = [\boldsymbol{\Phi}_{\mathrm{r}}(\mathbf{x})\ \ \boldsymbol{\Phi}_{\mathrm{j}}(\mathbf{x})]$ is an $m \times 2$ matrix, where $\boldsymbol{\Phi}_{\mathrm{r}}(\mathbf{x})$ and $\boldsymbol{\Phi}_{\mathrm{j}}(\mathbf{x})$ are its first and second column, respectively, and $\bar{\boldsymbol{\phi}}(\mathbf{x}) = \boldsymbol{\phi}^{\top}(\mathbf{x})$:

$$\mathbf{f}_{\mathbb{R}}(\mathbf{x}) = \begin{bmatrix} f_{\mathrm{r}}(\mathbf{x}) \\ f_{\mathrm{j}}(\mathbf{x}) \end{bmatrix} = \boldsymbol{\phi}^{\top}(\mathbf{x})\mathbf{w} = \begin{bmatrix} \boldsymbol{\Phi}_{\mathrm{r}}^{\top}(\mathbf{x}) \\ \boldsymbol{\Phi}_{\mathrm{j}}^{\top}(\mathbf{x}) \end{bmatrix} \mathbf{w}. \tag{5}$$

The objective is now the minimization of

$$J(\mathbf{w}) = \mathbb{E}\left[\left(\mathbf{y}_{\mathbb{R}}(i) - \boldsymbol{\phi}^{\top}(\mathbf{x}(i))\mathbf{w} \right)^{\top} \left(\mathbf{y}_{\mathbb{R}}(i) - \boldsymbol{\phi}^{\top}(\mathbf{x}(i))\mathbf{w} \right) \right]. \tag{6}$$

It is easy to show that the gradient is

$$\begin{aligned} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} &= -2\mathbb{E}\left[\boldsymbol{\phi}(\mathbf{x}(i)) \left(\mathbf{y}_{\mathbb{R}}(i) - \boldsymbol{\phi}^{\top}(\mathbf{x}(i))\mathbf{w} \right) \right] \\ &= -2\mathbb{E}[\boldsymbol{\phi}(\mathbf{x}(i))\mathbf{e}_{\mathbb{R}}(i)], \end{aligned} \tag{7}$$

and the update equation for $\mathbf{w}$ using the stochastic gradient yields

$$\mathbf{w}(i) = \mathbf{w}(i-1) + 2\mu\boldsymbol{\phi}(\mathbf{x}(i))\mathbf{e}_{\mathbb{R}}(i). \tag{8}$$

If we set $\mathbf{w}(0) = \mathbf{0}$, the repeated application of the weight-update equation (8) yields

$$\mathbf{w}(i) = 2\mu \sum_{l=1}^{i} \boldsymbol{\phi}(\mathbf{x}(l))\mathbf{e}_{\mathbb{R}}(l). \tag{9}$$

At instant i the output can be estimated using the last updated weights, $\mathbf{w}(i-1)$, as $\hat{\mathbf{y}}_{\mathbb{R}}(i) = \mathbf{f}_{\mathbb{R}}(\mathbf{x}(i)) = \boldsymbol{\phi}^{\top}(\mathbf{x}(i))\mathbf{w}(i-1)$. Therefore, the input-output operation of the composite KLMS algorithm can be expressed as

$$\begin{aligned} \hat{\mathbf{y}}_{\mathbb{R}}(i) &= \boldsymbol{\phi}^{\top}(\mathbf{x}(i))\mathbf{w}(i-1) \\ &= 2\mu \sum_{l=1}^{i-1} \boldsymbol{\phi}^{\top}(\mathbf{x}(i))\boldsymbol{\phi}(\mathbf{x}(l))\mathbf{e}_{\mathbb{R}}(l) \\ &= 2\mu \sum_{l=1}^{i-1} \mathbf{K}\left(\mathbf{x}(i), \mathbf{x}(l)\right) \mathbf{e}_{\mathbb{R}}(l), \end{aligned} \tag{10}$$

where the matrix-valued kernel yields:

$$\begin{aligned} \mathbf{K}(\mathbf{x}(i), \mathbf{x}(l)) &= \boldsymbol{\phi}^{\top}(\mathbf{x}(i))\boldsymbol{\phi}(\mathbf{x}(l)) \\ &= \begin{bmatrix} \boldsymbol{\Phi}_{\mathrm{r}}^{\top}(\mathbf{x}(i)) \\ \boldsymbol{\Phi}_{\mathrm{j}}^{\top}(\mathbf{x}(i) \end{bmatrix} \left[\boldsymbol{\Phi}_{\mathrm{r}}(\mathbf{x}(l)) \boldsymbol{\Phi}_{\mathrm{j}}(\mathbf{x}(l)) \right] \\ &= \begin{bmatrix} \boldsymbol{\Phi}_{\mathrm{r}}^{\top}(\mathbf{x}(i))\boldsymbol{\Phi}_{\mathrm{r}}(\mathbf{x}(l)) & \boldsymbol{\Phi}_{\mathrm{r}}^{\top}(\mathbf{x}(i))\boldsymbol{\Phi}_{\mathrm{j}}(\mathbf{x}(l)) \\ \boldsymbol{\Phi}_{\mathrm{j}}^{\top}(\mathbf{x}(i))\boldsymbol{\Phi}_{\mathrm{r}}(\mathbf{x}(l)) & \boldsymbol{\Phi}_{\mathrm{j}}^{\top}(\mathbf{x}(i))\boldsymbol{\Phi}_{\mathrm{j}}(\mathbf{x}(l)) \end{bmatrix} \\ &= \begin{bmatrix} k_{\mathrm{rr}}(\mathbf{x}(i), \mathbf{x}(l)) & k_{\mathrm{rj}}(\mathbf{x}(i), \mathbf{x}(l)) \\ k_{\mathrm{jr}}(\mathbf{x}(i), \mathbf{x}(l)) & k_{\mathrm{jj}}(\mathbf{x}(i), \mathbf{x}(l)) \end{bmatrix}. \end{aligned} \tag{11}$$

Notice that this kernel matrix follows the structure introduced in [11] for the WL-RKHS, and is composed of four scalar real functions.

IV. The proposed generalized complex KLMS algorithm

Any real-valued composite vector representation $\mathbf{y}_{\mathbb{R}} = [\mathbf{y}_{\mathrm{r}}^{\top}\ \mathbf{y}_{\mathrm{j}}^{\top}]^{\top} \in \mathbb{R}^{2n}$ of any complex-valued vector $\mathbf{y} = \mathbf{y}_{\mathrm{r}} + \mathrm{j}\mathbf{y}_{\mathrm{j}} \in \mathbb{C}^n$, can be related to the complex *augmented* vector $\underline{\mathbf{y}} = [\mathbf{y}^{\top}\ \mathbf{y}^{\mathrm{H}}]^{\top} \in \mathbb{C}^{2n}$ representation, which is obtained by stacking $\mathbf{y}$ on top of its complex conjugate $\mathbf{y}^*$. The relation is $\underline{\mathbf{y}} = \mathbf{T}_n\mathbf{y}_{\mathbb{R}}$, where

$$\mathbf{T}_n = \begin{bmatrix} \mathbf{I} & \mathrm{j}\mathbf{I} \\ \mathbf{I} & -\mathrm{j}\mathbf{I} \end{bmatrix} \in \mathbb{C}^{2n \times 2n}, \tag{12}$$

which is a unitary matrix up to a factor of 2: $\mathbf{T}_n\mathbf{T}_n^{\mathrm{H}} = \mathbf{T}_n^{\mathrm{H}}\mathbf{T}_n = 2\mathbf{I}$, where $\mathbf{I}$ is the identity matrix.

We can now apply this relation to (10) to calculate:

$$\begin{aligned} \underline{\hat{\mathbf{y}}}(i) = \begin{bmatrix} \hat{y}(i) \\ \hat{y}^*(i) \end{bmatrix} &= \mathbf{T}_1\hat{\mathbf{y}}_{\mathbb{R}}(i) = \mathbf{T}_1 2\mu \sum_{l=1}^{i-1} \mathbf{K}(\mathbf{x}(i), \mathbf{x}(l))\mathbf{e}_{\mathbb{R}}(l) \\ &= 2\mu \sum_{l=1}^{i-1} \mathbf{T}_1\mathbf{K}(\mathbf{x}(i), \mathbf{x}(l)) \left(\frac{1}{2}\mathbf{T}_1^{\mathrm{H}}\mathbf{T}_1 \right) \mathbf{e}_{\mathbb{R}}(l) \\ &= \mu \sum_{l=1}^{i-1} \mathbf{K}_A(\mathbf{x}(i), \mathbf{x}(l))\underline{\mathbf{e}}(l). \end{aligned} \tag{13}$$

Here we have the augmented error vector $\underline{\mathbf{e}}(l) = \mathbf{T}_1\mathbf{e}_{\mathbb{R}}(l) = [e(l)\ e^*(l)]^\top$, and the augmented kernel matrix

$$\begin{aligned}\mathbf{K}_A(\mathbf{x}(i),\mathbf{x}(l)) &= \mathbf{T}_1\mathbf{K}(\mathbf{x}(i),\mathbf{x}(l))\mathbf{T}_1^{\mathrm{H}} \\ &= \begin{bmatrix} k(\mathbf{x}(i),\mathbf{x}(l)) & \tilde{k}(\mathbf{x}(i),\mathbf{x}(l)) \\ \tilde{k}^*(\mathbf{x}(i),\mathbf{x}(l)) & k^*(\mathbf{x}(i),\mathbf{x}(l)) \end{bmatrix}, \end{aligned} \quad (14)$$

where by using (11) the complex kernel and complex pseudo-kernel can be identified, respectively, as

$$\begin{aligned} k(\mathbf{x}(i),\mathbf{x}(l)) &= k_{\mathrm{rr}}(\mathbf{x}(i),\mathbf{x}(l)) + k_{\mathrm{jj}}(\mathbf{x}(i),\mathbf{x}(l)) \\ &\quad + \mathrm{j}\left(k_{\mathrm{jr}}(\mathbf{x}(i),\mathbf{x}(l)) - k_{\mathrm{rj}}(\mathbf{x}(i),\mathbf{x}(l))\right), \quad (15) \\ \tilde{k}(\mathbf{x}(i),\mathbf{x}(l)) &= k_{\mathrm{rr}}(\mathbf{x}(i),\mathbf{x}(l)) - k_{\mathrm{jj}}(\mathbf{x}(i),\mathbf{x}(l)) \\ &\quad + \mathrm{j}\left(k_{\mathrm{jr}}(\mathbf{x}(i),\mathbf{x}(l)) + k_{\mathrm{rj}}(\mathbf{x}(i),\mathbf{x}(l))\right). \quad (16) \end{aligned}$$

Notice that this kernel and pseudo-kernel follow the structure introduced in [11].

The first entry of $\hat{\underline{\mathbf{y}}}(i)$ in (13) yields the proposed generalized complex KLMS (gCKLMS):

$$\hat{y}(i) = \mu\sum_{l=1}^{i-1} e(l)k(\mathbf{x}(i),\mathbf{x}(l)) + \mu\sum_{l=1}^{i-1} e^*(l)\tilde{k}(\mathbf{x}(i),\mathbf{x}(l)). \quad (17)$$

V. Connection with other algorithms

In [6], two realizations of the complex-valued KLMS (CKLMS) algorithm were developed by following two methodologies. The first approach is based on using a complex-valued kernel for a complex RKHS through the associated feature map. In this approach, denoted in [6] as CKLMS2, the output yields:

$$\hat{y}(i) = \mu\sum_{l=1}^{i-1} e(l)k(\mathbf{x}(i),\mathbf{x}(l)). \quad (18)$$

The second alternative is the *complexification* of real RKHSs. In this approach, the space of complex-valued functions $f(\mathbf{x}) = f_1(\mathbf{x}) + \mathrm{j}f_2(\mathbf{x})$ is defined from functions $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ that are in a RKHS of real functions with real kernel $k_{\mathcal{R}}$. Then, the complexified real kernel trick allows to construct a kernel adaptive algorithm denoted in [6] as CKLMS1:

$$\hat{y}(i) = \mu\sum_{l=1}^{i-1} 2e(l)k_{\mathcal{R}}(\mathbf{x}(i),\mathbf{x}(l)). \quad (19)$$

Notice that the kernel used in this CKLMS1 algorithm is a real-valued function.

In [14] the framework of [6] is applied to develop widely linear adaptive filters in complex RKHS. Two realizations of the augmented CKLMS (ACKLMS) were proposed. First, by using the complexification approach they obtain exactly the same formula (19) for the CKLMS1 algorithm (except for a rescaling) [14]. On the other hand, when a pure complex-valued kernel is used, the ACKLMS algorithm yields

$$\hat{y}(i) = \mu\sum_{l=1}^{i-1} \left(e(l)k(\mathbf{x}(i),\mathbf{x}(l)) + e(l)k^*(\mathbf{x}(i),\mathbf{x}(l))\right). \quad (20)$$

At this point it is interesting to note that (20) and (19) are the same. If we take $e(l)$ as a common factor in (20), it follows:

$$\begin{aligned}\hat{y}(i) &= \mu\sum_{l=1}^{i-1} e(l)\left(k(\mathbf{x}(i),\mathbf{x}(l)) + k^*(\mathbf{x}(i),\mathbf{x}(l))\right) \\ &= \mu\sum_{l=1}^{i-1} 2e(l)\mathcal{R}\left\{k(\mathbf{x}(i),\mathbf{x}(l)\right\}. \end{aligned} \quad (21)$$

Hence (20) and (19) provide the same learning process, since in both cases the kernel is real. In fact, they yield the same formula with $\mathcal{R}\{k\} = k_{\mathcal{R}}$.

Next, we show that algorithms CKLMS1, CKLMS2 [6], and ACKLMS [14] are particular cases of our proposed gCKLMS algorithm in (17). They yield a subset of the cases the gCKLMS algorithm presented in this paper can represent.

First, these approaches do not have a pseudo-kernel term, therefore they provide simplified limited versions and hence a reduction on the flexibility the general algorithm provides. It is easy to check that if we set the pseudo-kernel equal to zero in (17) the gCKLMS reduces to the CKLMS2 in (18). However, to have $\tilde{k}(\mathbf{x}(i),\mathbf{x}(l)) = 0$ in (16) the following conditions must be satisfied:

$$\begin{aligned} k_{\mathrm{rr}}(\mathbf{x}(i),\mathbf{x}(l)) &= k_{\mathrm{jj}}(\mathbf{x}(i),\mathbf{x}(l)), \\ k_{\mathrm{jr}}(\mathbf{x}(i),\mathbf{x}(l)) &= -k_{\mathrm{rj}}(\mathbf{x}(i),\mathbf{x}(l))), \end{aligned} \quad (22)$$

and the kernel in (15) for the CKLMS2 algorithm yields

$$k(\mathbf{x}(i),\mathbf{x}(l)) = 2k_{\mathrm{rr}}(\mathbf{x}(i),\mathbf{x}(l)) - \mathrm{j}2k_{\mathrm{rj}}(\mathbf{x}(i),\mathbf{x}(l)). \quad (23)$$

Second, if in addition to $\tilde{k}(\mathbf{x}(i),\mathbf{x}(l)) = 0$ we now set $k_{\mathrm{rj}}(\mathbf{x}(i),\mathbf{x}(l)) = 0$, then the kernel in (23) becomes a real-valued function $k(\mathbf{x}(i),\mathbf{x}(l)) = 2k_{\mathrm{rr}}(\mathbf{x}(i),\mathbf{x}(l))$, and the gCKLMS simplifies to the CKLMS1 in (19) or the ACKLMS in (21).

A. Kernel design

The conditions that the algorithms impose on the kernel and pseudo-kernel terms must be carefully analyzed in order to choose the best algorithm and kernels for a given learning problem.

The kernel in an RKHS learning algorithm encodes our assumptions about the function that is being learned [5] and provides a measure of similarity between the inputs. In [11] the kernel and pseudo-kernel in (15)-(16) are analyzed, and several remarks are provided to help to design them and to decide when they should be real or complex-valued. We use that analysis here to understand the implications of the conditions that each algorithm imposes.

We start with the conditions imposed when the pseudo-kernel is null, i.e., the conditions in (22) that yield the complex-valued kernel in (23). For any two inputs $\mathbf{x}$ and $\mathbf{x}'$, the first condition $k_{\mathrm{rr}}(\mathbf{x},\mathbf{x}') = k_{\mathrm{jj}}(\mathbf{x},\mathbf{x}')$ implies that the same measure of similarity must be used with the real and the imaginary parts of the function [11]. Hence, if we impose a null pseudo-kernel, we cannot use a kernel for the real part, $k_{\mathrm{rr}}(\mathbf{x},\mathbf{x}')$, and another different design for the imaginary part, $k_{\mathrm{jj}}(\mathbf{x},\mathbf{x}')$. The second condition is $k_{\mathrm{jr}}(\mathbf{x},\mathbf{x}') = -k_{\mathrm{rj}}(\mathbf{x},\mathbf{x}')$.

TABLE I
CONDITIONS IMPOSED ON THE KERNEL (15) AND PSEUDO-KERNEL (16) BY THE ALGORITHMS

Algorithm	Kernel	Pseudo-kernel	Conditions
gCKLMS	$k(\mathbf{x},\mathbf{x}') = k_{\text{rr}}(\mathbf{x},\mathbf{x}') + k_{\text{jj}}(\mathbf{x},\mathbf{x}') + \text{j}\left(k_{\text{jr}}(\mathbf{x},\mathbf{x}') - k_{\text{rj}}(\mathbf{x},\mathbf{x}')\right)$	$\tilde{k}(\mathbf{x},\mathbf{x}') = k_{\text{rr}}(\mathbf{x},\mathbf{x}') - k_{\text{jj}}(\mathbf{x},\mathbf{x}') + \text{j}\left(k_{\text{jr}}(\mathbf{x},\mathbf{x}') + k_{\text{rj}}(\mathbf{x},\mathbf{x}')\right)$	$k_{\text{rj}}(\mathbf{x},\mathbf{x}') = -k_{\text{rj}}(\mathbf{x}',\mathbf{x})$
CKLMS2	$k(\mathbf{x},\mathbf{x}') = 2k_{\text{rr}}(\mathbf{x},\mathbf{x}') - \text{j}2k_{\text{rj}}(\mathbf{x},\mathbf{x}')$	$\tilde{k}(\mathbf{x},\mathbf{x}') = 0$	$k_{\text{rr}}(\mathbf{x},\mathbf{x}') = k_{\text{jj}}(\mathbf{x},\mathbf{x}')$, $k_{\text{jr}}(\mathbf{x},\mathbf{x}') = -k_{\text{rj}}(\mathbf{x},\mathbf{x}')$
ACKLMS/CKLMS1	$k(\mathbf{x},\mathbf{x}') = 2k_{\text{rr}}(\mathbf{x},\mathbf{x}') \in \mathbb{R}$	$\tilde{k}(\mathbf{x},\mathbf{x}') = 0$	$k_{\text{rr}}(\mathbf{x},\mathbf{x}') = k_{\text{jj}}(\mathbf{x},\mathbf{x}')$, $k_{\text{jr}}(\mathbf{x},\mathbf{x}') = -k_{\text{rj}}(\mathbf{x},\mathbf{x}') = 0$

But we also have $k_{\text{jr}}(\mathbf{x},\mathbf{x}') = k_{\text{rj}}(\mathbf{x}',\mathbf{x})$, because the matrix-valued kernel $\mathbf{K}(\mathbf{x},\mathbf{x}')$ in (11) must be positive semi-definite (property (c) in Section II). Therefore, $k_{\text{rj}}(\mathbf{x},\mathbf{x}') = -k_{\text{rj}}(\mathbf{x}',\mathbf{x})$ and $k_{\text{rj}}(\mathbf{x},\mathbf{x}) = k_{\text{jr}}(\mathbf{x},\mathbf{x}) = 0$. This imposes a skew-symmetry in the measure of similarity between the real and the imaginary parts of the function.

As an example, the complex Gaussian kernel proposed in [6] for the CKLMS2 algorithm:

$$k_{\mathbb{C}G}(\mathbf{x},\mathbf{x}') = \exp\left(-(\mathbf{x}-\mathbf{x}'^{*})^{\top}(\mathbf{x}-\mathbf{x}'^{*})/\gamma_{\mathbb{C}G}^2\right), \qquad (24)$$

follows the form given in (23) and fulfils the conditions in (22), i.e., the symmetries that yield a null pseudo-kernel. However, this kernel measures similarities between the real parts of the inputs with $||\mathbf{x}_{\text{r}}-\mathbf{x}'_{\text{r}}||^2$, while for the imaginary ones it uses $||\mathbf{x}_{\text{j}}+\mathbf{x}'_{\text{j}}||^2$, where $||\cdot||$ is the ℓ^2-norm. Also, it is not stationary, has an oscillatory behavior, and the exponent in the kernel may easily grow large and positive [24]. This might cause numerical problems and, as we show later in the experiments, it does not yield the best performance.

As another example, in [19] it is proposed another complex-valued kernel for the CKLMS2 algorithm, the so-called independent kernel:

$$\begin{aligned} k_{\text{ind}}(\mathbf{x},\mathbf{x}') = {} & \kappa_{\mathbb{R}G}\left(\mathbf{x}_{\text{r}},\mathbf{x}'_{\text{r}}\right) + \kappa_{\mathbb{R}G}\left(\mathbf{x}_{\text{j}},\mathbf{x}'_{\text{j}}\right) \\ & + \text{j}\left(\kappa_{\mathbb{R}G}\left(\mathbf{x}_{\text{r}},\mathbf{x}'_{\text{j}}\right) - \kappa_{\mathbb{R}G}\left(\mathbf{x}_{\text{j}},\mathbf{x}'_{\text{r}}\right)\right), \end{aligned} \qquad (25)$$

where $\mathbf{x} = \mathbf{x}_{\text{r}} + \text{j}\mathbf{x}_{\text{j}}$ and $\kappa_{\mathbb{R}G}\left(\mathbf{z},\mathbf{z}'\right) = \exp\left(-||\mathbf{z}-\mathbf{z}'||^2/\gamma_{\mathbb{R}G}^2\right)$ is the well-known real Gaussian kernel of real inputs $\mathbf{z}$ and $\mathbf{z}'$. The CKLMS2 algorithm with this kernel was named the independent CKLMS estimate (iCKLMS). This independent kernel mimics the structure in (15), but it uses the same real function $\kappa_{\mathbb{R}G}$ for all the terms in (15), and the inputs to this function are not the complex-valued inputs $\mathbf{x}$, but their real or imaginary parts. Also, for the iCKLMS estimate the pseudo-kernel is null, hence the independent kernel has a skew-symmetric imaginary part, i.e., $\kappa_{\mathbb{R}G}(\mathbf{x}_{\text{r}},\mathbf{x}'_{\text{j}}) - \kappa_{\mathbb{R}G}(\mathbf{x}_{\text{j}},\mathbf{x}'_{\text{r}}) = -(\kappa_{\mathbb{R}G}(\mathbf{x}'_{\text{r}},\mathbf{x}_{\text{j}}) - \kappa_{\mathbb{R}G}(\mathbf{x}'_{\text{j}},\mathbf{x}_{\text{r}}))$.

When the pseudo-kernel is null, the skew-symmetry $k_{\text{rj}}(\mathbf{x},\mathbf{x}') = -k_{\text{rj}}(\mathbf{x},\mathbf{x}')$ in the imaginary part of the kernels used by the CKLMS2 or the iCKLSM algorithms may not be a property satisfied by many to-be-learned functions and, in such a case, enforcing a complex-valued kernel can be counterproductive. Algorithms CKLMS1 [6] and ACKLMS [14] avoid this problem by adding another condition: $k_{\text{rj}}(\mathbf{x}(i),\mathbf{x}(l)) = 0$. This means that these algorithms use a real-valued kernel of the form $k(\mathbf{x},\mathbf{x}') = 2k_{\text{rr}}(\mathbf{x},\mathbf{x}')$. The condition $k_{\text{rj}}(\mathbf{x}(i),\mathbf{x}(l)) = 0$ implies that the real and the imaginary parts are not related and that one of them does not provide information to learn the other [11].

In Table I we summarize, for every algorithm, the conditions imposed on the kernel and pseudo-kernel terms.

We conclude that algorithms CKLMS1, CKLMS2 [6], iCKLMS [19] and ACKLMS [14] cannot represent any possible complex-valued function, and yield a subset of the cases that the gCKLMS algorithm proposed in this paper can represent. This is the motivation to name our proposal 'generalized' CKLMS. The gCKLMS, with the kernel and the pseudo-kernel terms, provides more flexibility to model the learning problem by means of the four real-valued functions k_{rr}, k_{jj}, k_{rj} and k_{jr}. Hence, the gCKLMS will improve results if the conditions described above are not suitable for our learning problem. Some interesting scenarios are as follows. If the real and imaginary parts of the output are not independent, and also they have different properties in terms of similarity, then the gCKLMS with a kernel and a pseudo-kernel is needed. If the real and imaginary parts of the output are independent, $k_{\text{rj}} = k_{\text{jr}} = 0$, but they still have different properties in terms of similarity, $k_{\text{rr}} \neq k_{\text{jj}}$, then kernel and pseudo-kernel are needed. Finally, when the real and imaginary parts of the output are independent, and they both have the same properties in terms of similarity, the pseudo-kernel is null and a real-valued kernel should be used. In general, setting the pseudo-kernel to zero and enforcing a complex-valued kernel is counterproductive unless you identify, for the particular problem at hand, a skew-symmetry of the kind $k_{\text{rj}}(\mathbf{x},\mathbf{x}') = -k_{\text{rj}}(\mathbf{x},\mathbf{x}')$.

We end this discussion about the kernels by bringing here a suitable real-valued function for k_{rr}, k_{jj}, k_{rj} and k_{jr} proposed in [11]. This is the adaptation to complex-valued inputs of the real-valued Gaussian kernel:

$$k_G(\mathbf{x},\mathbf{x}') = \exp\left(-(\mathbf{x}-\mathbf{x}')^{\text{H}}(\mathbf{x}-\mathbf{x}')/\gamma^2\right), \qquad (26)$$

where γ is the kernel parameter. This real function provides a measure of similarity between the complex-valued inputs that is simple but effective for complex-valued signals: inputs closer to other input in the complex field are considered more similar than inputs that are further away [11]. We will use it in our experiments. For a further analysis about the selection of suitable kernels for complex-valued applications see [11], [24].

VI. EXPERIMENTS

We consider two experiments where we compare the performance of our proposal the gCKLMS in (17), versus the CKLMS2 in (18) [6], the iCKLMS [19], and the ACKLMS algorithm in (21) [14].

In the first experiment, we reproduce the nonlinear channel equalization task in [14]. In this experiment, the complex-valued signals have independent real and imaginary parts, and they are better represented with different kernels. We show that in such a case the best choice is a real kernel and a real pseudo-kernel.

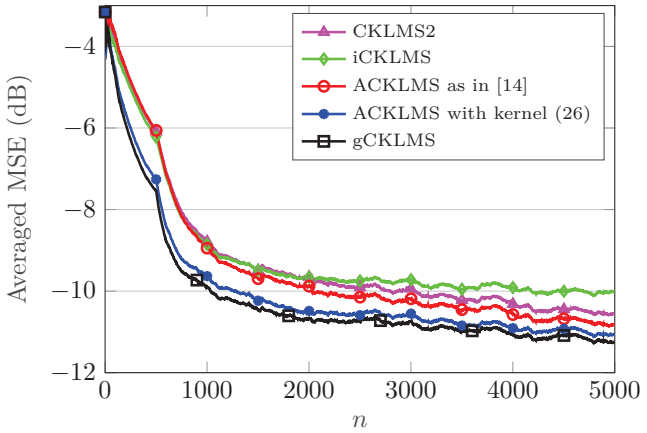

Fig. 1. MSE in dB versus the number of input samples for the *soft nonlinear channel* with the circular input case for the CKLMS2 ($\gamma_{\mathbb{C}G} = 10$), the iCKLMS ($\gamma_{\mathbb{R}G} = 5$), the ACKLMS as in [14] ($\gamma_{\mathbb{C}G} = 10$), the ACKLMS with kernel (26) ($\gamma_{\mathrm{r}} = 5$), and the gCKLMS ($\gamma_{\mathrm{r}} = 6.5$ and $\gamma_{\mathrm{j}} = 5.5$).

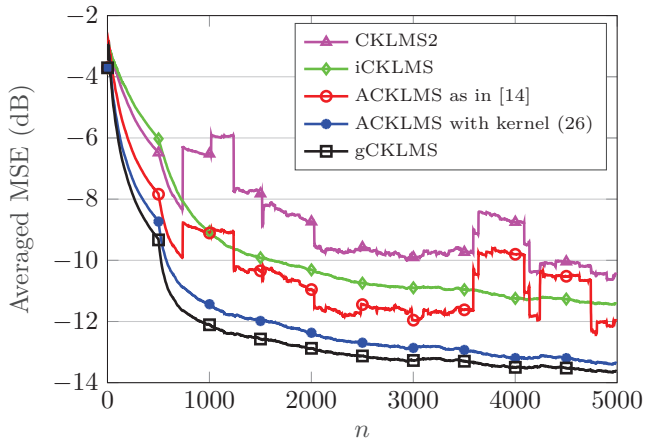

Fig. 2. MSE in dB versus the number of input samples for the *soft nonlinear channel* with the noncircular input case for the CKLMS2 ($\gamma_{\mathbb{C}G} = 10$), the iCKLMS ($\gamma_{\mathbb{R}G} = 5$), the ACKLMS as in [14] ($\gamma_{\mathbb{C}G} = 10$), the ACKLMS with kernel (26) ($\gamma_{\mathrm{r}} = 5$), and the gCKLMS ($\gamma_{\mathrm{r}} = 6.5$ and $\gamma_{\mathrm{j}} = 5.5$).

In the second experiment, we propose learning a filtered two-dimensional random process. At the output of the filter, the real and imaginary parts are not independent, and we show that we can use the imaginary part of the pseudo-kernel to improve the performance.

As in [14], we use the complex Gaussian kernel $k_{\mathbb{C}G}(\mathbf{x}, \mathbf{x}')$ in (24) for both the CKLMS2 and the ACKLMS. In fact, for the ACKLMS the real part of this kernel is used, as was shown in (21). We use the code available in [25] to run the algorithms. For the iCKLMS, the CKLMS2 in (18) with the independent kernel (25) is used.

For our proposed gCKLMS we use the general kernel and pseudo-kernel in (15)-(16). For $k_{\mathrm{rr}}(\mathbf{x}, \mathbf{x}')$, $k_{\mathrm{jj}}(\mathbf{x}, \mathbf{x}')$, $k_{\mathrm{jr}}(\mathbf{x}, \mathbf{x}')$ and $k_{\mathrm{rj}}(\mathbf{x}, \mathbf{x}')$ we propose to use the real-valued Gaussian kernel $k_G(\mathbf{x}, \mathbf{x}')$ in (26) with parameter $\gamma = \gamma_{\mathrm{r}}$ for k_{rr}, $\gamma = \gamma_{\mathrm{j}}$ for k_{jj}, $\gamma = \gamma_{\mathrm{rj}}$ for k_{rj}, and $\gamma = \gamma_{\mathrm{jr}}$ for k_{jr}, respectively. The kernel and pseudo-kernel can be simplified if the signals meet any of the conditions discussed in Section V-A. For example, if the real and imaginary parts of the signals are independent we can set $k_{\mathrm{jr}}(\mathbf{x}, \mathbf{x}') = k_{\mathrm{rj}}(\mathbf{x}, \mathbf{x}') = 0$ and the kernel and pseudo-kernel are real-valued:

$$k(\mathbf{x}, \mathbf{x}') = k_{\mathrm{rr}}(\mathbf{x}, \mathbf{x}') + k_{\mathrm{jj}}(\mathbf{x}, \mathbf{x}'), \tag{27}$$

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = k_{\mathrm{rr}}(\mathbf{x}, \mathbf{x}') - k_{\mathrm{jj}}(\mathbf{x}, \mathbf{x}'). \tag{28}$$

We use this simplification in the first experiment. Notice that if we also assume that the real and imaginary parts of the output use the same kernel, $k_{\mathrm{rr}} = k_{\mathrm{jj}}$, then we should set $\gamma_{\mathrm{r}} = \gamma_{\mathrm{j}}$ and the pseudo-kernel term in (28) cancels. In such a case, as explained in Section V, the gCKLMS approach simplifies to the ACKLMS with real-valued kernel $k(\mathbf{x}, \mathbf{x}') = 2k_{\mathrm{rr}}(\mathbf{x}, \mathbf{x}')$, where $k_{\mathrm{rr}}(\mathbf{x}, \mathbf{x}')$ is as in (26) with $\gamma = \gamma_{\mathrm{r}}$. We will refer to this case as ACKLMS with kernel (26) in the experiments.

A. Nonlinear channel equalization

We face the problem of nonlinear channel equalization, as in [6] and [14], for ease of comparison and continuity. The channel consists of a linear filter and a memoryless nonlinearity. The two nonlinear channels in [14] have been considered here. The first channel is the *soft nonlinear channel*, with linear

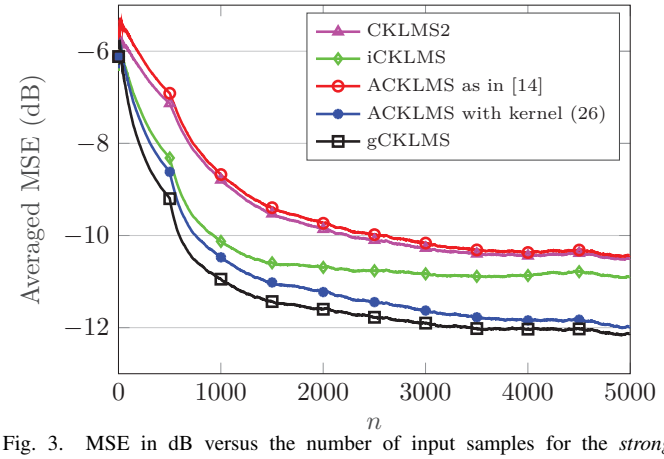

Fig. 3. MSE in dB versus the number of input samples for the *strong nonlinear channel* with the circular input case for the CKLMS2 ($\gamma_{\mathbb{C}G} = 15$), the iCKLMS ($\gamma_{\mathbb{R}G} = 5$), the ACKLMS as in [14] ($\gamma_{\mathbb{C}G} = 15$), the ACKLMS with kernel (26) ($\gamma_{\mathrm{r}} = 5$), and the gCKLMS ($\gamma_{\mathrm{r}} = 5$ and $\gamma_{\mathrm{j}} = 3$).

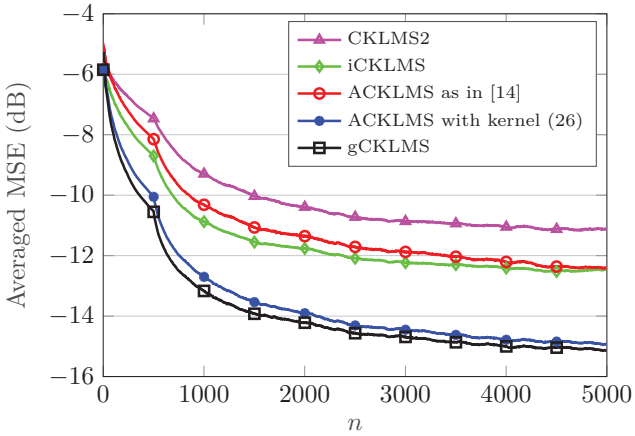

Fig. 4. MSE in dB versus the number of input samples for the *strong nonlinear channel* with the noncircular input case for the CKLMS2 ($\gamma_{\mathbb{C}G} = 15$), the iCKLMS ($\gamma_{\mathbb{R}G} = 5$), the ACKLMS as in [14] ($\gamma_{\mathbb{C}G} = 15$), the ACKLMS with kernel (26) ($\gamma_{\mathrm{r}} = 5$), and the gCKLMS ($\gamma_{\mathrm{r}} = 5$ and $\gamma_{\mathrm{j}} = 3$).

filter

$$t(n) = (-0.9 + 0.8\mathrm{j}) \cdot s(n) + (0.6 - 0.7\mathrm{j}) \cdot s(n-1),$$

followed by the nonlinearity

$$q(n) = t(n) + (0.1 + 0.15\mathrm{j}) \cdot t^2(n) + (0.06 + 0.05\mathrm{j}) \cdot t^3(n).$$

The second one is the *strong nonlinear channel*, with linear filter

$$\begin{aligned} t(n) =& (-0.9 + 0.8\mathrm{j}) \cdot s(n) + (0.6 - 0.7\mathrm{j}) \cdot s(n-1) \\ &+ (-0.4 + 0.3\mathrm{j}) \cdot s(n-2) + (0.3 - 0.2\mathrm{j}) \cdot s(n-3) \\ &+ (-0.1 - 0.2\mathrm{j}) \cdot s(n-4), \end{aligned}$$

and nonlinearity

$$q(n) = t(n) + (0.2 + 0.25\mathrm{j}) \cdot t^2(n) + (0.08 + 0.09\mathrm{j}) \cdot t^3(n).$$

At the receiver, the signal $q(n)$ is corrupted by additive white circular Gaussian noise with an SNR of 15 dB to yield the received signal $r(n)$. The inputs to the equalizer are the sets of samples $\mathbf{x} = [r(n+D), r(n+D-1), \cdots, r(n+D-L+1)]^\top$, where $L > 0$ is the filter length and D is the equalization time delay. Here we set $L = 5$ and $D = 2$, as in [14]. The goal is to estimate the original input signal $s(n)$.

1) Gaussian distributed inputs: We first set the input signals as in [14]: $s(n) = 0.7(\sqrt{1-\rho^2} \cdot X(n) + \mathrm{j}\rho \cdot Y(n))$, where $X(n)$ and $Y(n)$ are independent Gaussian random variables, with $\rho = 1/\sqrt{2}$ for circular signals, and $\rho = 0.1$ for noncircular signals.

The real and imaginary parts of the signals are independent and, therefore, we can set $k_{\mathrm{jr}}(\mathbf{x}, \mathbf{x}') = k_{\mathrm{rj}}(\mathbf{x}, \mathbf{x}') = 0$ and use the real-valued kernel and pseudo-kernel terms in (27)-(28) for our proposed gCKLMS.

Experiments were conducted on 100 independent sets of 5000 samples of the input signal. For all the approaches, the novelty criterion [26], [27], is used for sparsification with $\delta_1 = 0.15$ and $\delta_2 = 0.2$, as in [14].

Figs. 1 and 2 show the averaged mean square errors (MSE) for the *soft nonlinear channel*. The circular input case is shown in Fig. 1, and the noncircular input case is shown in Fig. 2. Since we are reproducing here the experiment in [14], for the CKLMS2 and the ACKLMS algorithms we set $\gamma_{\mathbb{C}G} = 10$ and $\mu = 1/8$, i.e., the same values used in [14]. For the iCKLMS we set $\gamma_{\mathbb{R}G} = 5$ and $\mu = 1/8$. For the gCKLMS algorithm we set $\gamma_{\mathrm{r}} = 6.5$ and $\gamma_{\mathrm{j}} = 5.5$, and $\mu = 1/7$. For the ACKLMS with kernel (26) we set $\gamma_{\mathrm{r}} = 5$ and $\mu = 1/10$.

Figs. 3 and 4 include the MSE for the *strong nonlinear channel* and the circular input and noncircular input cases, respectively. Again, since we are reproducing the experiment in [14], for the CKLMS2 and the ACKLMS algorithms we set $\gamma_{\mathbb{C}G} = 15$ and $\mu = 1/6$, i.e., the same values used in [14]. For the iCKLMS we set $\gamma_{\mathbb{R}G} = 5$ and $\mu = 1/8$. For the gCKLMS algorithm we set $\gamma_{\mathrm{r}} = 5$ and $\gamma_{\mathrm{j}} = 3$, and $\mu = 1/7$. For the ACKLMS with kernel (26) we set $\gamma_{\mathrm{r}} = 5$ and $\mu = 1/10$.

In all the examples, the proposed gCKLMS outperforms the other algorithms. The main advantage of the gCKLMS is that by introducing a pseudo-kernel we can use a different kernel for the real and the imaginary parts, $k_{\mathrm{rr}}(\mathbf{x}, \mathbf{x}')$ and $k_{\mathrm{jj}}(\mathbf{x}, \mathbf{x}')$. This extra degree of freedom, which is not present in the other algorithms, is the key to obtain a better estimation. In this experiment, the gain in MSE is small, because $k_{\mathrm{rr}}(\mathbf{x}, \mathbf{x}')$ and $k_{\mathrm{jj}}(\mathbf{x}, \mathbf{x}')$ are very similar. That is the reason why the ACKLMS with kernel (26), i.e., setting $\gamma_{\mathrm{r}} = \gamma_{\mathrm{j}}$, performs well and close to the general case. In any case, it can be observed that to achieve a given error, the faster convergence of the gCKLMS allows saving 10%-30% of the samples and time. The dictionary size after sparsification and the running times of the tested algorithms are in Table II and III[1].

With the complex Gaussian kernel, both the ACKLMS in [14] and the CKLM2 in [6] perform poorly compared to the gCKLMS. Therefore, the experiments show that the complex Gaussian kernel is not the best choice for this equalization task and, as it is shown in Fig. 2, sometimes yields undesired spikes in the learning curves.

TABLE II
DICTIONARY SIZE AFTER SPARSIFICATION AND RUNNING TIMES (IN SECONDS) FOR THE SOFT NONLINEAR CHANNEL AND GAUSSIAN DISTRIBUTED INPUTS

	Circular Input		Noncircular Input	
Algorithm	Dict. size	Run time (s)	Dict. size	Run time (s)
gCKLMS	2966	4.6181	1866	3.5118
CKLMS2	3291	4.6314	2692	4.1555
ACKLMS [14]	3173	4.4803	2102	3.7701
ACKLMS, kernel (26)	3038	3.8185	1926	3.2533
iCKLMS [19]	3419	5.5193	2692	3.3111

TABLE III
DICTIONARY SIZE AFTER SPARSIFICATION AND RUNNING TIMES (IN SECONDS) FOR THE STRONG NONLINEAR CHANNEL AND GAUSSIAN DISTRIBUTED INPUTS

	Circular Input		Noncircular Input	
Algorithm	Dict. size	Run time (s)	Dict. size	Run time (s)
gCKLMS	2565	5.1858	1439	3.0639
CKLMS2	3222	6.3349	2565	3.5103
ACKLMS [14]	3259	5.3047	2220	3.4936
ACKLMS, kernel (26)	2698	4.7781	1528	2.6946
iCKLMS [19]	2958	4.7952	2312	4.2762

2) Unbalanced digital modulated signals: In digital communications inputs are discrete. For discrete and unbalanced digital modulated signals, the difference between $k_{\mathrm{rr}}(\mathbf{x}, \mathbf{x}')$ and $k_{\mathrm{jj}}(\mathbf{x}, \mathbf{x}')$ is greater and the proposed gCKLMS algorithm is a good choice versus the previous proposals, with null pseudo-kernels. To illustrate this point, we propose to repeat here the equalization experiment for the *soft nonlinear channel*, where the input signals are now $s(n) = 0.2X(n) + \mathrm{j}0.1Y(n)$, where $X(n)$ and $Y(n)$ are independent binary $\{-1, +1\}$ data streams.

For the proposed gCKLMS algorithm we use again the real-valued Gaussian kernel (26) with parameters $\gamma_{\mathrm{r}} = 0.59$ for $k_{\mathrm{rr}}(\mathbf{x}, \mathbf{x}')$ and $\gamma_{\mathrm{j}} = 1.63$ for $k_{\mathrm{jj}}(\mathbf{x}, \mathbf{x}')$, respectively. For the ACKLMS with kernel (26) we set $\gamma_{\mathrm{r}} = 1.52$. The learning parameter is set to $\mu = 0.5$ for both approaches.

We generate 100 independent test trials with a set of 10000 samples to test the algorithms. The mean square errors (MSE) of the estimation are compared in Fig. 5 versus the number of input samples. It can be observed that the proposed gCKLMS

[1]Matlab 2018a on a computer with Windows Server 2016, 64 GB RAM and Intel Xeon 2.80 GHz Processor

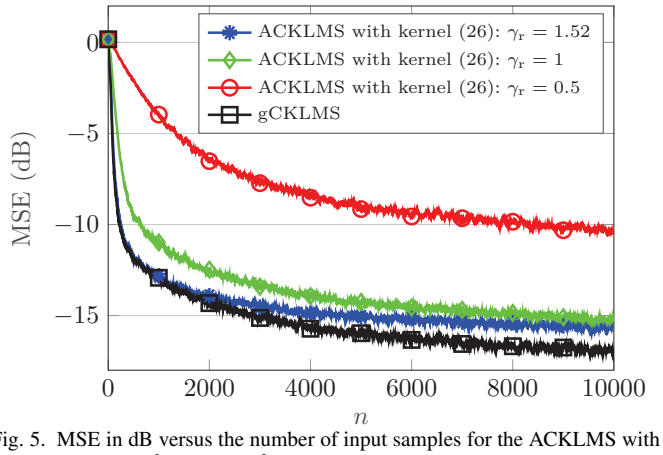

Fig. 5. MSE in dB versus the number of input samples for the ACKLMS with kernel (26) ($\gamma_r = \{0.5, 1, 1.52\}$) and the gCKLMS ($\gamma_r = 0.59$, $\gamma_j = 1.63$).

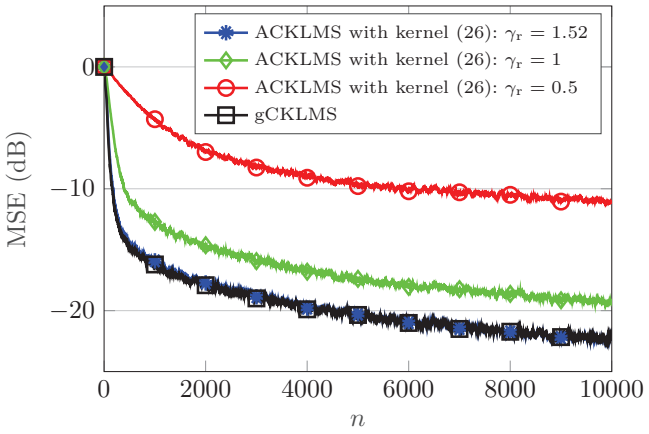

Fig. 6. MSE in dB for the imaginary part versus the number of input samples for the ACKLMS with kernel (26) ($\gamma_r = \{0.5, 1, 1.52\}$) and the gCKLMS ($\gamma_r = 0.59$, $\gamma_j = 1.63$).

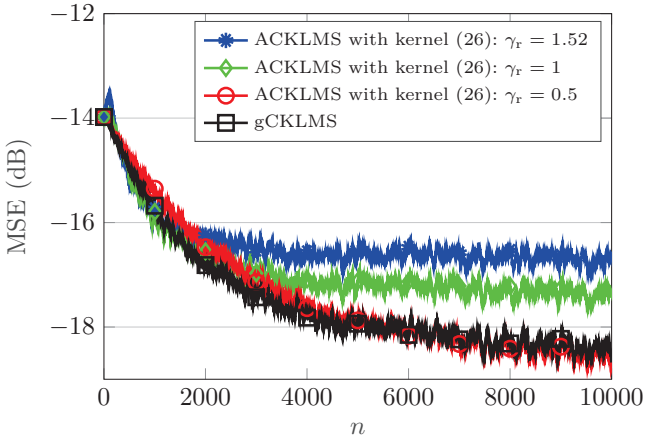

Fig. 7. MSE in dB for the real part versus the number of input samples for the ACKLMS with kernel (26) ($\gamma_r = \{0.5, 1, 1.52\}$) and the gCKLMS ($\gamma_r = 0.59$, $\gamma_j = 1.63$).

outperforms the ACKLMS with kernel (26), i.e., the case with null pseudo-kernel.

Again, the key to obtaining a better estimation with the gCKLMS in this experiment is the possibility to define a different kernel for the real part $k_{rr}(\mathbf{x}, \mathbf{x}')$ and the imaginary part $k_{jj}(\mathbf{x}, \mathbf{x}')$. Figs. 6 and 7 are included to highlight this. Fig. 6 shows the estimation MSE of the imaginary part of the signals, while Fig. 7 shows the estimation MSE of the real part. In this experiment the real and imaginary parts require a different kernel to be accurately learnt. However, the ACKLMS algorithm uses the same kernel for both parts. The parameter value γ_r with the best performance to learn the imaginary part of the output in Fig. 6 yields the worst estimation of the real part in Fig. 7. And vice versa, the best parameter value to learn the real part of the output in Fig. 7 provides the worst performance in Fig. 6. Remarkably, the estimation with the proposed gCKLMS is always low for both imaginary and real parts, as it allows to set different values for $k_{rr}(\mathbf{x}, \mathbf{x}')$ and $k_{jj}(\mathbf{x}, \mathbf{x}')$.

B. A random process filtered

In this experiment, we show the performance of the proposed gCKLMS when the signals do not have independent real and imaginary parts. We generate a complex-valued signal with correlated real and imaginary parts by filtering a real-valued random process with a complex-valued filter. We define the complex-valued filter $h(x) = \alpha \cdot \left(2\exp(-|x|^2/3) + \mathrm{j}\exp(-|x|^2/0.5\right)$, where $x = x_r + \mathrm{j}x_j$, with $x_r \in [-5, 5]$ and $x_j \in [-5, 5]$, and $\alpha = 0.0228$ to ensure unit norm. Then we define a real Gaussian process $s(x_r, x_j)$ with zero mean and unit variance, and we pass this process through the filter. We show in Figs. 8 and 9 the real and imaginary parts of one sample of the filtered process in $x_r \in [-5, 5]$ and $x_j \in [-5, 5]$.

At the output of the filter we get a Gaussian process. There is a known relationship between the variance of a Gaussian process and the kernel, and between the pseudo-covariance of the Gaussian process and the pseudo-kernel [8]. Therefore, the covariance and pseudo-covariance of the process help in the design of suitable kernels. It can be shown that the covariance of the process at the output of the filter is real-valued with two additive terms. Hence, we can conclude that for the gCKLMS we could use a kernel with the following terms:

$$k(\mathbf{x}, \mathbf{x}') = k_{rr}(\mathbf{x}, \mathbf{x}') + k_{jj}(\mathbf{x}, \mathbf{x}'). \tag{29}$$

Also, it can be found that the pseudo-covariance of the process is complex-valued, and this is an indication that there exists a correlation between the real and the imaginary parts of the filtered process [1]. Thus, we can conclude that for the gCKLMS we could use a pseudo-kernel with the following terms:

$$\tilde{k}(\mathbf{x}, \mathbf{x}') = k_{rr}(\mathbf{x}, \mathbf{x}') - k_{jj}(\mathbf{x}, \mathbf{x}') + 2\mathrm{j}k_{jr}(\mathbf{x}, \mathbf{x}'). \tag{30}$$

All the terms in the covariance and pseudo-covariance are exponentials of the form $\exp(-d_x^* d_x/\gamma)$, where $d_x = x - x'$ [8], having different γ values and scalings. Accordingly, we propose to use the real-valued Gaussian kernel $k_G(\mathbf{x}, \mathbf{x}')$ in (26) for $k_{rr}(\mathbf{x}, \mathbf{x}')$, $k_{jj}(\mathbf{x}, \mathbf{x}')$ and $k_{jr}(\mathbf{x}, \mathbf{x}')$, with different scaling and kernel parameters. Further details about the construction of kernels based on covariances can be found in [8].

In the gCKLMS algorithm, the real and imaginary parts of the error e provide two degrees of freedom to adjust the amplitudes of the kernel terms as the learning process

advances. As we now have a third term, we use an extra amplitude parameter v to provide another degree of freedom to allow each term of the kernels to have a different scaling. Hence, for $k_{\mathrm{rr}}(\mathbf{x},\mathbf{x}')$ and $k_{\mathrm{jj}}(\mathbf{x},\mathbf{x}')$ we use the real-valued Gaussian kernel with parameters γ_{r} and γ_{j}, respectively. For the imaginary part of the pseudo-kernel, we also use the real-valued Gaussian kernel with parameter γ_{jr} and scaled with v, i.e., $k_{\mathrm{jr}}(\mathbf{x},\mathbf{x}') = v \cdot k_G(\mathbf{x}',\mathbf{x}')$. In a training stage we set $\gamma_{\mathrm{r}} = 1.73$, $\gamma_{\mathrm{j}} = 0.58$, $\gamma_{\mathrm{jr}} = 1.11$ and $v = 0.09$. The learning step was set to $\mu = 1/4$.

For the ACKLMS we again propose $k(\mathbf{x},\mathbf{x}') = 2k_{\mathrm{rr}}(\mathbf{x},\mathbf{x}')$, with $k_{\mathrm{rr}}(\mathbf{x},\mathbf{x}')$ the real-valued Gaussian kernel $k_G(\mathbf{x},\mathbf{x}')$ in (26), since it yields better performance than the complex Gaussian kernel. The kernel parameter is set to $\gamma_{\mathrm{r}} = 0.76$ and $\mu = 1/2$.

We generate 100 independent samples of the filtered process in $x_{\mathrm{r}} \in [-5,5]$ and $x_{\mathrm{j}} \in [-5,5]$, each with 10000 data points. A random white circular Gaussian noise is added to the samples. The averaged MSE of the estimation are compared in Fig. 10 versus the number of input points for two values of SNR, 15 and 50 dB. The proposed gCKLMS algorithm greatly outperforms the ACKLMS algorithm. We also include in the figure the performance of the gCKLMS when $v = 0$, i.e., when both the kernel and pseudo-kernel are real-valued. Fig. 10 shows that the imaginary part of the pseudo-kernel helps to improve the prediction accuracy by making use of the $k_{\mathrm{jr}}(\mathbf{x},\mathbf{x}')$ term in the pseudo-kernel.

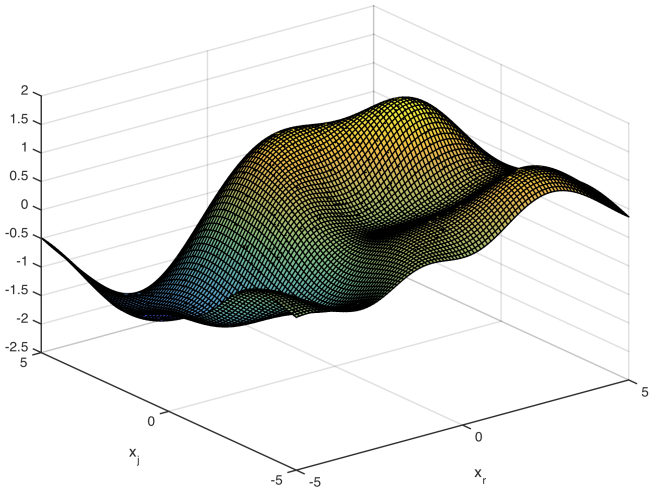

Fig. 8. Real part of a sample of the filtered process.

VII. Conclusions

In this paper, we have developed a novel generalized formulation for the complex-valued KLMS algorithm. Based on the ideas recently presented in [11] for the WL-RKHS, we have developed the gCKLMS algorithm that includes both a kernel and a pseudo-kernel. We reviewed the theory of RKHS of vector-valued functions to define the feature map for the RKHS of composite vector-valued functions. Based on this definition, we were able to develop the composite KLMS algorithm to later rewrite it in augmented notation and, finally, yield the proposed gCKLMS algorithm. Also, in this process, we were able to identify the equations that define the kernel and pseudo-kernel. These equations follow the structure introduced in [11], and include four real-valued functions: $k_{\mathrm{rr}}(\mathbf{x},\mathbf{x}')$, $k_{\mathrm{jj}}(\mathbf{x},\mathbf{x}')$, $k_{\mathrm{rj}}(\mathbf{x},\mathbf{x}')$ and $k_{\mathrm{jr}}(\mathbf{x},\mathbf{x}')$. We can use the analysis in [11] to design these real-valued functions and set the kernel and pseudo-kernel for a given application. Another important contribution of the paper is to show that previous proposed complex-valued KLMS algorithms are just particular simplifications of the gCKLMS proposed in this paper. The experiments included reveal that the gain of using the gCKLMS algorithm, which provides more flexibility than the previously proposed algorithms, can be significant.

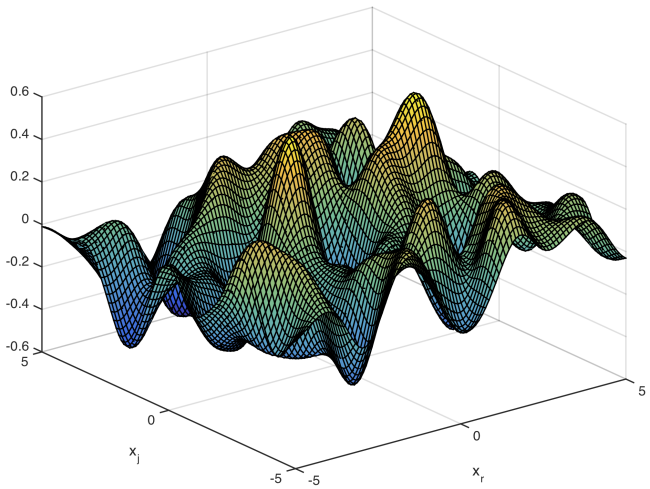

Fig. 9. Imaginary part of a sample of the filtered process.

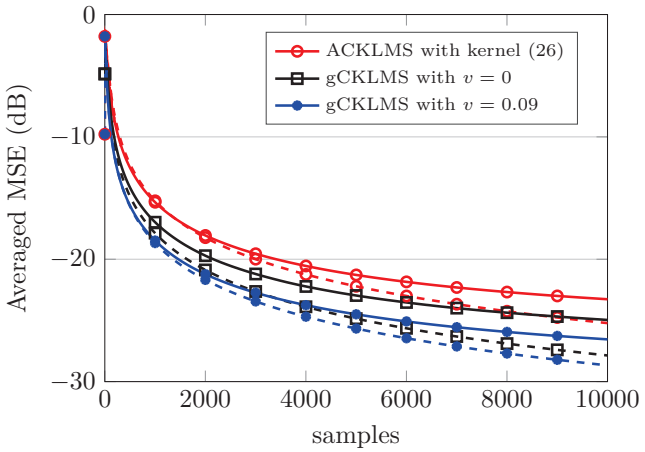

Fig. 10. Averaged MSE in dB versus the number of input samples for the ACKLMS ($\gamma_{\mathrm{r}} = 0.76$), the gCKLMS ($\gamma_{\mathrm{r}} = 1.73$, $\gamma_{\mathrm{j}} = 0.58$, $\gamma_{\mathrm{jr}} = 1.11$) with $v = 0$ and $v = 0.09$. Dashed lines: SNR = 50 dB. Solid lines: SNR = 15 dB.

References

[1] P. J. Schreier and L. L. Scharf, *Statistical Signal Processing of Complex-Valued Data. The Theory of Improper and Noncircular Signals*. Cambridge, UK: Cambridge University Press, 2010.

[2] A. Hirose, *Complex-Valued Neural Networks: Advances and Applications*, ser. IEEE Press Series on Computational Intelligence. Wiley, 2013.

[3] M. E. Valle, "Complex-valued recurrent correlation neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 9, pp. 1600–1612, Sept 2014.

[4] D. Mandic and V. S. L. Goh, *Complex Valued Nonlinear Adaptive Filters: Noncircularity, Widely Linear and Neural Models*. Wiley Publishing, 2009.

[5] B. Schölkopf and A. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, ser. Adaptive computation and machine learning. Cambridge, Massachusetts: MIT Press, 2002.

[6] 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.

[7] S. Van 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, Aug 2012.

[8] R. Boloix-Tortosa, J. J. Murillo-Fuentes, F. J. Payán-Somet, and F. Pérez-Cruz, "Complex Gaussian processes for regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5499–5511, Nov 2018.

[9] A. Papaioannou and S. Zafeiriou, "Principal component analysis with complex kernel: The widely linear model," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 9, pp. 1719–1726, Sept 2014.

[10] I. Steinwart, D. Hush, and C. Scovel, "An explicit description of the reproducing kernel Hilbert spaces of Gaussian RBF kernels," *IEEE Trans. Inf. Theory*, vol. 52, no. 10, pp. 4635–4643, Oct 2006.

[11] R. Boloix-Tortosa, J. J. Murillo-Fuentes, I. Santos, and F. Pérez-Cruz, "Widely linear complex-valued kernel methods for regression," *IEEE Trans. Signal Process.*, vol. 65, no. 19, pp. 5240–5248, Oct 2017.

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

[13] T. Ogunfunmi and T. K. Paul, "On the complex kernel-based adaptive filter." in *IEEE Int. Symp. on Circuits and Systems, ISCAS*, 2011, pp. 1263–1266.

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

[15] B. Picinbono and P. Chevalier, "Widely linear estimation with complex data," *IEEE Trans. on Signal Processing*, vol. 43, no. 8, pp. 2030–2033, Aug 1995.

[16] C. A. Micchelli and M. Pontil, "On learning vector-valued functions," *Neural Computation*, vol. 17, no. 1, pp. 177–204, 2005.

[17] A. Kuh and D. Mandic, "Applications of complex augmented kernels to wind profile prediction," in *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2009, pp. 3581–3584.

[18] A. Kuh, C. Manloloyo, R. Corpuz, and N. Kowahl, "Wind prediction using complex augmented adaptive filters," in *The 2010 International Conference on Green Circuits and Systems*, June 2010, pp. 46–50.

[19] F. Tobar, A. Kuh, and D. Mandic, "A novel augmented complex valued kernel LMS," in *IEEE 7th Sensor Array and Multichannel Signal Processing Workshop, SAM*, 2012, pp. 473–476.

[20] N. Aronszajn, "Theory of reproducing kernels," *Trans. Amer. Math. Soc.*, vol. 68, pp. 337–404, 1950.

[21] M. A. Álvarez, L. Rosasco, and N. D. Lawrence, "Kernels for vector-valued functions: A review," *Found. Trends Mach. Learn.*, vol. 4, no. 3, pp. 195–266, Mar. 2012.

[22] A. Caponnetto, C. A. Micchelli, M. Pontil, and Y. Ying, "Universal multi-task kernels," *J. Mach. Learn. Res.*, vol. 9, pp. 1615–1646, Jun. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1390681.1442785

[23] B. Widrow, J. McCool, and M. Ball, "The complex LMS algorithm," *Proc. IEEE*, vol. 63, no. 4, pp. 719–720, April 1975.

[24] R. Boloix-Tortosa, F. J. Payán-Somet, E. A. de Reyna, and J. J. Murillo-Fuentes, "Complex kernels for proper complex-valued signals: a review," in *2015 23rd European Signal Processing Conference (EUSIPCO)*, Aug 2015, pp. 2371–2375.

[25] P. Bouboulis, "The Augmented Complex Kernel LMS Matlab and C Code," http://bouboulis.mysch.gr/kernels.html, [Online].

[26] W. Liu, J. C. Principe, and S. Haykin, *Kernel Adaptive Filtering: A Comprehensive Introduction*, 1st ed. Wiley Publishing, 2010.

[27] J. Platt, "A resource-allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, June 1991.