1. INTRODUCTION

Single-kernel adaptive filters have been extensively studied over the last decade, and their performance has been investigated experimentally and theoretically on a variety of real-valued nonlinear system identification problems. Typical filtering algorithms in reproducing kernel Hilbert spaces (RKHS) are the KRLS algorithm [1], the sliding-window KRLS algorithm [2], and the quantized KRLS algorithm [3]. The KNLMS algorithm was independently introduced in [4–7]. The KLMS algorithm, proposed in [8, 9], has attracted much attention in recent years because of its simplicity and robustness. An analysis of its convergence behavior with Gaussian kernels is reported in [10], and a closed-form condition for convergence is introduced in [11]. The stability of this algorithm with I-norm regularization is studied in [12, 13].

Kernel-based adaptive filtering algorithms for complex data have recently attracted attention since they ensure phase processing. This is of importance for applications in communication, radar and sonar. A complexified kernel LMS algorithm and pure complex kernel LMS algorithm are introduced in [14]. A direct extension of the derivations in [10] is proposed in [15] to analyze the convergence behavior of complex KLMS algorithm (CKLMS). The augmented CKLMS algorithm (ACKLMS) is presented in [16, 17], and its normalized counterpart is described in [18, 19]. These works show that augmented complex-valued algorithms provide significantly improved performance compared with complex-valued algorithms. Finally, the quaternion KLMS algorithm has been recently introduced in [20] as an extension of complex-valued KLMS algorithms.

The aim of this paper is to analyze the convergence behavior of the ACKLMS algorithm. First, we introduce some definitions and a general framework for pure complex multikernel adaptive filtering algorithms. This framework relies on multikernel adaptive filters that has previously been derived for use with real-valued data in [21–24]. Then, we derive models for the convergence behavior in the mean and mean-square sense of the ACKLMS algorithm with Gaussian kernels. Finally, the accuracy of these models is checked with simulation results.

2. COMPLEX MULTI-KERNEL LMS

2.1. Preliminaries

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

\[
\begin{align*}
    u(n) &= \sqrt{1 - \rho^2} u_{re}(n) + i \rho u_{im}(n) \\
    &= u_{re}(n) + i u_{im}(n)
\end{align*}
\]

(1)

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

Let \( \kappa : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{C} \) be a complex reproducing kernel. We denote by \( (\cdot, \cdot)_{\kappa} \) the induced complex RKHS with its inner product. Complex reproducing kernels include the Szegő kernel, the Bergman kernel, and the so-called pure complex Gaussian kernel. The latter is the extension of the Gaussian kernel for complex arguments. The pure complex Gaussian kernel is defined as follows [25]

\[
\kappa_{\mathbb{C}}(u, v) = \exp \left( -\sum_{\ell=1}^{L} (u_{\ell} - v_{\ell})^2/2\xi^2 \right)
\]

(2)

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

\[
\kappa_{\mathbb{C}}^{*}(u, v) = \exp \left( -\sum_{\ell=1}^{L} (u_{\ell}^* - v_{\ell}^*)^2/2\xi^2 \right)
\]

(3)

Note that \( (\cdot)^{*} \) is defined on kernels and should not be confused with the complex conjugate \( (\cdot)^{\ast} \). We shall focus on the above complex Gaussian kernel in the sequel.
2.2. A framework for complex multi-kernel algorithms

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

\[
\Phi : \mathbb{C} \rightarrow \mathbb{C}^K, \\
\mathbf{u} \mapsto \Phi(\mathbf{u}) = \text{col}\{\varphi_1(\mathbf{u}), \ldots, \varphi_K(\mathbf{u})\} \tag{4}
\]

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

\[
(\Phi, \Phi')_{\mathcal{H}} = \sum_{k=1}^K (\varphi_k, \varphi_k')_{\mathbb{H}_k}. \tag{5}
\]

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

\[
\Phi(\mathbf{u}) = [\Phi, \kappa_{\mathcal{H}}(\cdot, \mathbf{u})] \tag{6}
\]

with \( \kappa_{\mathcal{H}}(\cdot, \mathbf{u}) = \text{col}\{\kappa_{C,1}(\cdot, \mathbf{u}), \ldots, \kappa_{C,K}(\cdot, \mathbf{u})\} \) and \( \langle \cdot, \cdot \rangle \) the entry-wise inner product. This yields the following reproducing property

\[
\kappa_{\mathcal{H}}(\cdot, \mathbf{u}, \mathbf{v}) = [\kappa_{\mathcal{H}}(\cdot, \mathbf{u}), \kappa_{\mathcal{H}}(\cdot, \mathbf{v})]. \tag{7}
\]

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

Given a valued input-output sequence \( \{(d(n), \mathbf{u}(n))\}_{n=1}^N \), we aim at estimating a multidimensional function \( \Psi \) in \( \mathcal{H} \) that minimizes the regularized least-square error

\[
\min_{\Psi \in \mathcal{H}} J(\Psi) = \sum_{n=1}^N |d(n) - \sum_{k=1}^K \kappa_{C,k}(\cdot, \mathbf{u}(n))|^2 + \lambda \| \sum_{k=1}^K \kappa_{C,k}(\cdot, \mathbf{u}(n)) \|_2^2 \tag{8}
\]

with \( \lambda \geq 0 \) a regularization constant. By virtue of the generalized multidimensional representer theorem, not presented in this paper due to lack of space, the optimum function \( \Psi \) can be written as

\[
\Psi(\cdot) = \text{col}\{\sum_{k=1}^K \alpha_{n,k} \kappa_{C,k}(\cdot, \mathbf{u}(n))\}_{k=1}^K. \tag{9}
\]

For simplicity, without loss of generality, we shall omit the regularization term in problem (8), which can be reformulated as

\[
\min_{\alpha} J(\alpha) = \sum_{n=1}^N |d(n) - \sum_{k=1}^K \alpha_{n,k} \kappa_{C,k}(\cdot, \mathbf{u}(n))|^2 \tag{10}
\]

where \( \alpha \) is the unknown weight vector, and \( \kappa_{C,k}(\cdot, \mathbf{u}(n)) \) is the \( N \times 1 \) kernelized input vector with \( j \)-th entry \( \kappa_{C,k}(\cdot, \mathbf{u}(j), \mathbf{u}(n)) \). Calculating the directional derivative of \( J(\alpha) \) with respect to \( \alpha \) by Wirtinger’s calculus yields

\[
\partial_{\alpha_n} J(\alpha) = -2 \sum_{n=1}^N e^n(n) \kappa_{C,k}(\cdot, \mathbf{u}(n)). \tag{11}
\]

where \( e(n) = d(n) - \sum_{k=1}^K \alpha_{n,k} \kappa_{C,k}(\cdot, \mathbf{u}(n)) \). Approximating (11) by its instantaneous estimate \( \partial_{\alpha_n} J(\alpha) \approx -2 e^n(n) \kappa_{C,k}(\cdot, \mathbf{u}(n)) \), we obtain the stochastic gradient descent algorithm:

\[
\alpha(n+1) = \alpha(n) + \eta e^n(n) \kappa_{\mathcal{H}}(n) = \sum_{i=1}^n \eta e^i(i) \kappa_{\mathcal{H}}(i) \tag{12}
\]

with \( \eta \) a positive step-size, \( \kappa_{\mathcal{H}}(n) = \text{col}\{\kappa_{C,k}(n)\}_{k=1}^K \) the complex kernelized input vector, and \( e(n) = d(n) - \alpha^H(n) \kappa_{\mathcal{H}}(n) \) the estimation error. Finally, the optimal function is of the form

\[
\psi(\cdot) = \sum_{n=1}^N \sum_{k=1}^K \alpha_{n,k}^* \kappa_{C,k}(\cdot, \mathbf{u}(n)). \tag{13}
\]

2.3. Augmented complex kernel LMS (ACKLMS)

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

\[
\psi(\cdot) = \sum_{m=1}^M \sum_{k=1}^K \alpha_{m,k} \kappa_{C,k}(\cdot, \mathbf{u}(\omega_m)) \tag{14}
\]

where \( \omega \triangleq \{\kappa_{\mathcal{H}}(\cdot, \mathbf{u}(\omega_m))\}_{m=1}^M \) is the so-called dictionary of the filter \( \psi \), and \( M \) its length. Limiting the number of single-kernel filters to \( K = 2 \), and setting the two kernels to (2), (3), the ACKLMS algorithm based on model (14) is given by (See [18] for an introduction to ACKLMS):

\[
d(n) = \sum_{m=1}^M \left[ \alpha_{1,m}^*(n) \kappa_C(n, \mathbf{u}(n)) + \alpha_{2,m}^*(n) \kappa'_C(n, \mathbf{u}(n)) \right] + \alpha_{2,m}^*(n) \kappa'_C(n, \mathbf{u}(n), \mathbf{u}(\omega_m)) \tag{15}
\]

The ACKLMS algorithm can be viewed as a complex Gaussian bi-kernel case of the complex multi-kernel algorithm [18, 19]. It can be expected that ACKLMS algorithm outperforms the existing CKLMS algorithms due to the flexibility of complex multi-kernels.

3. ACKLMS PERFORMANCE ANALYSIS

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

\[
E\left\{ |e(n)|^2 | \omega \right\} = \int_{\omega \in \mathbb{C}} |e(n)|^2 d\rho(\mathbf{u}(n), d(n) | \omega), \tag{16}
\]

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

\[
e_{\omega}(n) = d(n) - \hat{d}_{\omega}(n) \tag{17}
\]

with \( \hat{d}_{\omega}(n) = d(n) - \hat{d}_{\omega}(n) \). Multiplying \( e_{\omega}(n) \) by its conjugate and taking the expected value yields the mean-square-error (MSE)

\[
J_{\text{MSE},\omega} = E\left\{ |d(n)|^2 \right\} - 2 \text{Re} \left( p_{\text{ed,}\omega} \alpha_{\omega}(n) \right) \tag{18}
\]

with \( R_{\omega} = E\left\{ \kappa_{\mathcal{H}}(\mathbf{u}(\omega_m)) \kappa_{\mathcal{H}}^*(\mathbf{u}(\omega_m)) | \omega \right\} \) the correlation matrix of input data, and \( p_{\text{ed,}\omega} = E\left\{ \kappa_{\mathcal{H}}(\mathbf{u}(\omega_m)) \hat{d}(n) | \omega \right\} \) the cross-correlation vector between \( \kappa_{\mathcal{H},\omega}(n) \) and \( d(n) \). As \( R_{\omega} \) is positive definite, the optimum weight vector is given by

\[
\alpha_{\omega,\text{opt}} = \arg \min_{\alpha_{\omega}} J_{\text{MSE},\omega}(\alpha_{\omega}) = R_{\omega}^{-1} p_{\text{ed,}\omega} \tag{19}
\]

and the minimum MSE is

\[
J_{\text{MSE},\omega} = E\left\{ |d(n)|^2 \right\} - p_{\text{ed,}\omega} R_{\omega}^{-1} p_{\text{ed,}\omega}. \tag{20}
\]
3.1. Mean weight error analysis

The weight update of the ACKLMS algorithm is given by

\[ \alpha_\omega(n+1) = \alpha_\omega(n) + \eta \epsilon_\omega(n) \kappa_{\|\omega\|}(n). \]  

(21)

Let \( v_\omega(n) \) be the weight error vector defined as

\[ v_\omega(n) = \alpha_\omega(n) - \alpha_{\text{opt},\omega}. \]  

(22)

The weight error vector update equation is then given by

\[ v_\omega(n+1) = v_\omega(n) + \eta \epsilon_\omega(n) \kappa_{\|\omega\|}(n). \]  

(23)

The error (17) is consequently rewritten as

\[ \epsilon_\omega(n) = d(n) - \kappa_{\|\omega\|}(n)v_\omega(n) - \kappa_{\|\omega\|}(n)\alpha_{\text{opt},\omega}. \]  

(24)

Substituting (24) into (23) yields

\[ v_\omega(n+1) = v_\omega(n) + \eta (d(n)\kappa_{\|\omega\|}(n)) - \kappa_{\|\omega\|}(n)v_\omega(n) - \kappa_{\|\omega\|}(n)\alpha_{\text{opt},\omega} \]

(25)

Taking expected value of (25), using the CMIA hypothesis introduced in [26], and [19], we get the mean weight error model:

\[ E\{v_\omega(n+1)\} = (I - \eta R_{\|\omega\|})E\{v_\omega(n)\}. \]  

(26)

The \((i,j)\)-th entry of matrix \( R_{\|\omega\|} \) is given by

\[ [R_{\|\omega\|}]_{i,j} = E\{k_{\|\omega\|}(u(n),u(\omega_i))k_{\|\omega\|}(u(n),u(\omega_j))\} \]  

(27)

with the complex Gaussian bi-kernel \( k_{\|\omega\|}(u(n),u(\omega_m)) \) given by

\[ k_{\|\omega\|}(u(n),u(\omega_m)) = \begin{cases} k_C(u(n),u(\omega_m)), & 1 \leq m \leq M \\ k^*_C(u(n),u(\omega_m)), & M+1 \leq m \leq 2M \end{cases} \]

(28)

Let us define a new vector that separates the real and imaginary parts of \( u(n) \) such that \( \tilde{u}(n) = \text{col}\{u_j(n),u_Q(n)\} \in \mathbb{R}^{2M} \). With the Gaussian kernels (2)-3), the expected value of (27) can be obtained by making use of the moment generating function in [26]. We get (28) where \( \delta_m \) is the indicator function

\[ \delta_m = \begin{cases} 1, & 1 \leq m \leq M \\ -1, & M+1 \leq m \leq 2M \end{cases} \]  

(29)

and \( \kappa_\omega = E\{\tilde{u}(n)\tilde{u}^T(n)\} \). The definition of \( H(i,j) \) in (28) depends on \( i \) and \( j \) as follows:

\[ H(i,j) = \begin{pmatrix} I & O \\ O & -I \end{pmatrix}, 1 \leq i, j \leq M \text{ and } M+1 \leq i, j \leq 2M \]

\[ H(i,j) = \begin{pmatrix} I & 1iI \\ 1iI & -I \end{pmatrix}, 1 \leq i \leq M \text{ and } M+1 \leq i \leq j \leq 2M \]

\[ H(i,j) = \begin{pmatrix} I & -1iI \\ -1iI & -I \end{pmatrix}, 1 \leq j \leq M \text{ and } M+1 \leq i \leq 2M \]

Vector \( b \) in (28) is given by

\[ b = \begin{pmatrix} -\sum_{s \in (i,j)} u_1(\omega_s) + 1i[\delta_i u_Q(\omega_i) - \delta_j u_Q(\omega_j)] \\ -\sum_{s \in (i,j)} u_Q(\omega_s) + 1i[\delta_i u_1(\omega_i) + \delta_j u_1(\omega_j)] \end{pmatrix}. \]  

(30)

Equation (26) leads to the following theorem (without proof due to lack of space):

**Theorem 3.1** (Stability in the mean) Assume CMIA introduced in [26] holds. Then, for any initial condition, given a dictionary \( \omega \), the Gaussian ACKLMS algorithm (21) asymptotically converges in mean if the step size is chosen to satisfy

\[ 0 < \eta < 2/\lambda_{\text{max}}(R_{\|\omega\|}) \]  

(31)

where \( \lambda_{\text{max}}(\cdot) \) denotes the maximum eigenvalue of its matrix argument. The entries of \( R_{\|\omega\|} \) are given by (28).

3.2. Mean-square error analysis

Using (24) and CMIA, MSE is related to the second-order moment of the weight vector by [10]

\[ J_{\text{MSE},\omega}(n) = J_{\text{min},\omega} + \text{trace}\{R_{\|\omega\|}C_{\omega,\omega}(n)\} \]  

(32)

where \( C_{\omega,\omega}(n) = E\{v_\omega(n)v_\omega^T(n)\} \) is the autocorrelation matrix of the weight error vector \( v_\omega(n) \), and \( J_{\text{min},\omega} \) is the minimum MSE given by (20). The analysis of the MSE behavior (32) requires a recursive model for \( C_{\omega,\omega}(n) \). Post-multiplying (25) by its Hermitian conjugate, taking the expected value, and using CMIA, we get the following recursion for sufficiently small step sizes

\[ C_{\omega,\omega}(n+1) \approx C_{\omega,\omega}(n) \]

\[ -\eta [R_{\|\omega\|}C_{\omega,\omega}(n) + C_{\omega,\omega}(n)R_{\|\omega\|}] \]

(33)

\[ + \eta^2 T_\omega(n) + \eta^2 R_{\|\omega\|}J_{\text{min},\omega} \]

with

\[ T_\omega(n) = E\{k_{\|\omega\|}(\tilde{u}(n),\tilde{u}(\omega))^2\} \]  

(34)

Evaluating (34) is a significant step in the analysis since \( k_{\|\omega\|}(\cdot) \) is a nonlinear transformation of a quadratic form of \( u(n) \). Using CMIA to determine the \((i,j)\)-th element of \( T_\omega(n) \) in (34) yields

\[ [T_\omega(n)]_{i,j} \approx \sum_{\ell=1}^{M} \sum_{p=1}^{M} E\{k_{\|\omega\|}(u_\ell(n),u(\omega_i))k_{\|\omega\|}(u_\ell(n),u(\omega_j))\}^* \times k_{\|\omega\|}(u(n),u(\omega_k))k_{\|\omega\|}(u(n),u(\omega_p)) \]  

(35)

This expression can be written as

\[ [T_\omega(n)]_{i,j} \approx \text{trace}\{K_{\omega}(i,j)C_{\omega,\omega}(n)\} \]  

(36)

where the \((\ell,p)\)-th entry of the matrix \( K_{\omega}(i,j) \) is given by

\[ [K_{\omega}(i,j)]_{\ell,p} = E\{k_{\|\omega\|}(u_\ell(n),u(\omega_i))k_{\|\omega\|}(u_\ell(n),u(\omega_j))\}^* \times k_{\|\omega\|}(u(n),u(\omega_k))k_{\|\omega\|}(u(n),u(\omega_p)) \].

(37)

Similarly, we also rewrite (37) in terms of vector \( \tilde{u}(n) \) and use the moment generating function [26]. This leads to (38)-(39). The definition of \( L(i,j) \) in (38) depends on \( i \) and \( j \) as follows:

\[ L(i,j) = \begin{pmatrix} 2I & O \\ O & -2I \end{pmatrix}, \begin{cases} 1 \leq i, j, \ell \leq M \\ i, j \leq M \text{ and } M+1 \leq \ell, p \leq 2M \\
\end{cases} \]

\[ L(i,j) = \begin{pmatrix} 2I & 1iI \\ 1iI & -2I \end{pmatrix}, \begin{cases} 1 \leq j \leq M \text{ and } M+1 \leq i \leq 2M \\ i, j \leq M \text{ and } M+1 \leq \ell, p \leq 2M \\
\end{cases} \]

\[ L(i,j) = \begin{pmatrix} 2I & 1iI \\ 1iI & -2I \end{pmatrix}, \begin{cases} 1 \leq j \leq M \text{ and } M+1 \leq i \leq 2M \\ i, j \leq M \text{ and } M+1 \leq \ell, p \leq 2M \\
\end{cases} \]
\[ [R_{n,ω}]_{i,j} = I + \frac{2}{\xi^2} H(i,j) R_{n,ω}^{-1} \cdot \exp \left( -\frac{1}{2\xi^2} \sum_{s=(i,j)} \| u_s(ω_s) \|^2 - \sum_{s=(i,j)} \| u_s(ω_s) \|^2 \right) \]
\times \exp \left( \frac{1}{\xi^2} \left[ \delta_i u_i^T(ω_i) u_Q(ω_i) - \delta_j u_j^T(ω_j) u_Q(ω_j) + \delta_i u_i^T(ω_i) u_Q(ω_i) - \delta_p u_p^T(ω_p) u_Q(ω_p) \right] \right) \exp \left( \frac{1}{2\xi^2} b^T R_n(I + \frac{2}{\xi^2} H(i,j) R_n)^{-1} b \right) \]
\[ \begin{bmatrix} K_{ω}(i,j) \end{bmatrix} = \left[ I + \frac{2}{\xi^2} L(i,j) R_{n,ω} \right] \cdot \frac{1}{\xi^2} \exp \left( \frac{1}{\xi^2} \left[ \delta_i u_i^T(ω_i) u_Q(ω_i) - \delta_j u_j^T(ω_j) u_Q(ω_j) + \delta_i u_i^T(ω_i) u_Q(ω_i) - \delta_p u_p^T(ω_p) u_Q(ω_p) \right] \right) \exp \left( \frac{1}{2\xi^2} f^T R_n(I + \frac{2}{\xi^2} L(i,j) R_n)^{-1} f \right) \]
\[ f = \left( -\sum_{s=(i,j,p)} u_s(ω_s) + 1i [\delta_i u_i(ω_i) - \delta_j u_j(ω_j) + \delta_i u_i(ω_i) - \delta_p u_p(ω_p)] \right) \]

\[ L(i,j) = \begin{pmatrix} 2I & -1iI \\ 1iI & -2I \end{pmatrix} \]
\[ L(i,j) = \begin{pmatrix} 2I & 2iI \\ 2iI & -2I \end{pmatrix} \]
\[ L(i,j) = \begin{pmatrix} 2I & -2iI \\ -2iI & -2I \end{pmatrix} \]

3.3. Steady-State behavior

In order to determine the steady-state of recursion (33), we rewrite it in a lexicographic form. Let \( \{ \cdot \} \) denote the operator that stacks the columns of a matrix on top of each other. Vectorizing \( C_{ω}(n) \) and \( R_{n,ω} \) by \( c_{ω}(n) = \text{vec} \{ C_{ω}(n) \} \) and \( r_{n,ω} = \text{vec} \{ R_{n,ω} \} \), we can rewrite (33) as follows

\[ c_{ω}(n) = G_{ω} c_{ω}(n) + \eta^2 J_{\text{min},ω} r_{n,ω} \]  

(41)

with \( G_{ω} = I - \eta (G_{ω,1} + G_{ω,2}) + \eta^2 G_{ω,3} \). Matrix \( G_{ω} \) is found by the use of the following definitions:

- \( I \) is the identity matrix of dimension \( 4M^2 \times 4M^2 \);
- \( G_{ω,1} = I \otimes R_{n,ω} \), where \( \otimes \) denotes the Kronecker product;
- \( G_{ω,2} = R_{n,ω} \otimes I \);
- \( G_{ω,3} \) is given by \( [G_{ω,3}]_{i+2(j-1)M+2(p-1)M} = [K_{ω}(i,j)]_{i,p} \) with \( 1 \leq i, j, p \leq 2M \).

Assuming convergence, the closed-formed solution of the recursion (41) in steady-state is given by

\[ c_{ω}(∞) = \eta^2 J_{\text{min},ω} (I - G_{ω})^{-1} r_{n,ω}. \]  

(42)

From equation (32), the steady-state MSE is finally given by

\[ J_{\text{MSE},ω}(∞) = J_{\text{min},ω} + \text{trace} \{ R_{n,ω} C_{ω,ω}(∞) \} \]  

(43)

where the second term on the right side is the steady-state EMSE.

4. EXPERIMENT

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

\[ u(n) = \rho u(n - 1) + \sigma u \sqrt{1 - \rho^2} \w(n) \]  

(44)

with \( w(n) = \sqrt{1 - \rho^2} \w(n) + i \rho \w(n) \). Parameter \( \rho \) was set to 0.1 corresponding to highly non-circular, and the random variables \( \w(n) \) and \( \w(n) \) were distributed according zero-mean i.i.d. Gaussian distributions with standard deviation \( \sigma_w = 1 \). Both parameters \( \rho_0 \) and \( \sigma_w \) were set to 0.5. The system to be identified was

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

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

\[ R_n = \sigma_u^2 \begin{pmatrix} 1 - \rho^2 & 1 - \rho^2 \\ 1 - \rho^2 & 1 - \rho^2 \end{pmatrix} \]  

(45)

The pure complex Gaussian bandwidth \( \xi \) and the step-size \( \eta \) were set to 0.55 and 0.1, respectively. We used the coherence sparsification criterion proposed in [5] with threshold \( \rho_{\text{th}} = 0.3 \) to construct a fixed dictionary of length \( M = 12 \). All simulation curves were obtained by averaging over 200 Monte Carlo runs. It is shown in Figure 1 that the theoretical curves consistently agree with the Monte Carlo simulations in both transient and steady-state.

5. CONCLUSION

In this paper, we presented the ACKLMS algorithm based on the framework of complex multi-kernel. Then we derived a theoretical model of convergence for ACKLMS with pre-tuned dictionary. In future works, we will study how using this model to design dictionaries, and set the step-size and the kernel bandwidth, that allow to reach specified MSE or convergence speed.
6. REFERENCES


