DISTRIBUTED MULTICHANNEL ADAPTIVE FILTERING

Fernando G. Almeida Neto, Vítor H. Nascimento[*]

University of São Paulo
São Paulo, Brazil
e-mail:{fganeto, vitor}@lps.usp.br

Amanda de Paula

Federal Rural University of Pernambuco
Cabo de Santo Agostinho, Brazil
e-mail: amanda.paula@uacsa.ufrpe.br

ABSTRACT

A new distributed multichannel technique is proposed for networks in which a different set of parameters is estimated by each node. The technique is proposed for non fully-connect topologies, so that nodes must store data and re-transmit information to other network elements. To reduce the amount of terms stored by each node, pre-computation of the data required by other elements is performed before the data sharing. The proposed method is adequate for implementation in networks with a large number of nodes, for which straightforward implementations would be prohibitive in terms of cost and memory.

Index Terms— Distributed adaptive filtering, distributed multichannel adaptive filtering, multichannel LMS

1. INTRODUCTION

Distributed adaptive filtering techniques implement decentralized processing by exploiting information shared by the elements which constitute the network. For this purpose, a group of agents is linked together by some connection topology and collaborates to estimate a set of parameters of interest. These techniques have been proposed for applications such as sensor networks and target detection [1, 2]. Recently, distributed networks were also proposed to model or mimic the behavior of biological networks, such as fish schools, bird formations and bee swarms [3–5]. Different strategies can be used to connect the agents (which are also called the nodes of the network), such as the incremental [6], the consensus [2, 7] and the diffusion [8–10] strategies. Regardless of the approach, in general these methods use locally obtained data and information shared by other nodes to estimate a set of parameters, which is the same for all nodes.

Despite the increasing number of publications on this subject, the techniques aforementioned cannot be applied to networks in which each node estimates a different set of parameters. For these networks, the development of algorithms is challenging and, to the best of our knowledge, is still an open problem. In this paper, we address this problem, and we propose distributed multichannel adaptive algorithms to exploit the collaboration between the nodes and improve the estimation of local parameters.

We start showing that a multichannel adaptive filter can be interpreted as a fully-connected, delayless-distributed network (DDN). Each channel of the adaptive filter corresponds to a node, with its own input data and set of parameters to estimate. The estimation performed in each node employs local data and data shared by other network elements, which must be available at the same time instant for all nodes. It is shown that each element can transmit its set of estimated coefficients and input data, allowing the other nodes to compute the quantities they need, or can perform pre-computation of the data required by their neighbors before the transmission. The pre-computation reduces the amount of transmitted data, saving energy in the communication links. However, the approach used by the fully-connected distributed multichannel adaptive filter may not be applied to other topologies, since non fully connected networks may have propagation delays, or to large networks, in which the amount of memory required in each node would become prohibitive. To extend the distributed method to other topologies, we develop techniques in which the nodes are allowed to store information obtained from other elements, so that these data can be re-transmitted to other agents latter. Using this scheme, many communication links can be removed from the network. Additionally, we show that the amount of information stored in each node can be reduced, if each node pre-computes the terms required by other elements before the transmission of data. Since the ideal number of elements stored in the nodes increases with the cardinality of the network, we describe an approximate scheme that limits the maximum amount of memory required in each node. The idea is to disregard the data shared by elements with low contribution to the estimation performed in node i. We present a simulation to support this approach.

Notation: Lower case is used for scalar quantities (e.g.: a) and bold lower case for column vectors (e.g.: $\mathbf{b}$). Bold capital letters represent matrices (e.g.: $\mathbf{A}$). diag$(\cdot)$ defines a diagonal matrix with the diagonal entries defined by the argument of the function, while $(\cdot)^T$ stands for the transposition of a matrix or vector. We use $\otimes$ to indicate the Kronecker product [11]. $||\cdot||_p$ denotes the p-norm.

[*]This work is partially funded by Fapesp and CNPq grants.

2. THE MULTICHANNEL FILTER VIEWED AS A DELAYLESS DISTRIBUTED NETWORK

In order to show how a multichannel adaptive filter can be interpreted as a distributed multichannel technique implemented as a DDN, consider the K-channel LMS algorithm (K-Ch-LMS). Assume that an N-entry regressor vector $\mathbf{x}_k(n)$, $k = 1, 2, \ldots, K$, is available for each channel at instant n. Additionally, consider that the vector $\mathbf{d}(n)$ of desired signals is modeled by

$$\mathbf{d}(n) = (\mathbf{W}^o)^T \boldsymbol{\chi}(n) + \mathbf{v}(n), \tag{1}$$

where $\mathbf{W}^o$ is the $KN \times K$ matrix of optimum coefficients, $\boldsymbol{\chi}(n) = \text{col}(\mathbf{x}_1(n),\ \mathbf{x}_2(n), \ldots,\ \mathbf{x}_K(n))$ stacks up the regressor vectors and $\mathbf{v}(n)$ is the noise vector, uncorrelated with $\boldsymbol{\chi}(n)$. Define

$$\mathbf{W}^o = \begin{bmatrix} \mathbf{w}^o_{1,1} & \cdots & \mathbf{w}^o_{1,K} \\ \vdots & \ddots & \vdots \\ \mathbf{w}^o_{K,1} & \cdots & \mathbf{w}^o_{K,K} \end{bmatrix},$$

where each $\mathbf{w}^o_{j,k}$, $j, k = 1, 2, \ldots, K$ is an N-length column vector. The k-th entry of $\mathbf{d}(n)$ is given by

$$d_k(n) = \sum_{j=1}^{K} (\mathbf{w}^o_{j,k})^T \mathbf{x}_j(n) + v_k(n), \tag{2}$$

so that the regressor vector of each channel contributes to the computation of the k-th entry of $\mathbf{d}(n)$.

The K-Ch-LMS algorithm estimates the matrix

$$\mathbf{W}(n) = \begin{bmatrix} \mathbf{w}_{1,1}(n) & \cdots & \mathbf{w}_{1,K}(n) \\ \vdots & \ddots & \vdots \\ \mathbf{w}_{K,1}(n) & \cdots & \mathbf{w}_{K,K}(n) \end{bmatrix}$$

to compute an approximation to $d_k(n)$ as given by

$$\hat{d}_k(n) = \sum_{j=1}^{K} \theta_{j,k}(n), \tag{3}$$

where $\theta_{j,k}(n) = \mathbf{w}^T_{j,k}(n)\mathbf{x}_j(n)$. The iterative update of $\mathbf{W}(n)$ uses the estimation error of each channel

$$e_k(n) = d_k(n) - \hat{d}_k(n) \tag{4}$$

to update the vectors of coefficients with

$$\mathbf{w}_{i,k}(n+1) = \mathbf{w}_{i,k}(n) + \mu e_k(n)\mathbf{x}_i(n), \tag{5}$$

$\forall i, k = 1, \ldots K$, where μ is the LMS step-size [12,13], which for simplicity we assume constant across all channels. Substituting eq. (3) in (4) and using the resulting expression in (5), one gets

$$\mathbf{w}_{i,k}(n+1) = \mathbf{w}_{i,k}(n) + \mu\left(d_k(n) - \sum_{j=1}^{K}\theta_{j,k}(n)\right)\mathbf{x}_i(n), \tag{6}$$

for $\forall i, k = 1, 2, \ldots K$. Notice from eq. (6) that the update of each $\mathbf{w}_{i,k}(n)$ depends on other entries of matrix $\mathbf{W}(n)$ and also on $\mathbf{x}_k(n)$, $k = 1, 2, \ldots, K$, so that one can say that the channels *collaborate* among them, *sharing* their input vectors and estimated coefficients $\mathbf{w}_{i,k}(n)$, to compute the updates. This can be interpreted as a characteristic of a distributed network, as we show now.

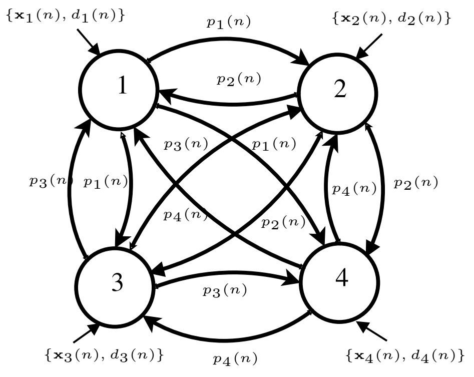
To interpret the K-Ch-LMS as a distributed network, assume that each channel from the K-Ch-LMS corresponds to a node. Consider that the i-th node measures $\mathbf{x}_i(n)$ and $d_i(n)$, and receives data from other nodes (here generically identified by $p_j(n)$, where j identifies the node that transmitted the data) to estimate $\mathbf{w}_{i,k}(n)$ (for $k = 1, 2, \ldots, K$). There is no propagation delay between the nodes, such that all information required to compute the estimates at time instant n is instantaneously available. Figure 1 shows the implementation of the four-channel LMS algorithm as a fully-connected, distributed delayless network.

In Fig. 1, node 1 estimates $\mathbf{w}_{1,k}(n+1)$, for $k = 1, 2, 3, 4$. Consider, for instance, the update of $\mathbf{w}_{1,2}(n+1)$. Recalling eq. (6), it requires $d_2(n)$, $\theta_{1,2}(n) = \mathbf{w}^T_{1,2}(n)\mathbf{x}_1(n)$, $\theta_{2,2}(n) = \mathbf{w}^T_{2,2}(n)\mathbf{x}_2(n)$, $\theta_{3,2}(n) = \mathbf{w}^T_{3,2}(n)\mathbf{x}_3(n)$ and $\theta_{4,2}(n) = \mathbf{w}^T_{4,2}(n)\mathbf{x}_4(n)$. Since only $\mathbf{w}_{1,1}(n)$ and $\mathbf{x}_1(n)$ are available at node 1, the remaining data must be obtained from the other nodes. One possible way for nodes 2, 3 and 4 to transmit their data to node 1 is the following: node 2 transmits $d_2(n)$, $\mathbf{w}_{2,2}(n)$ and $\mathbf{x}_2(n)$, while nodes 3 and 4 share $\mathbf{w}_{3,2}(n)$ and $\mathbf{x}_3(n)$, and $\mathbf{w}_{4,2}(n)$ and $\mathbf{x}_4(n)$, respectively, allowing node 1 to locally compute $\theta_{2,2}(n)$, $\theta_{3,2}(n)$ and $\theta_{4,2}(n)$. The amount of data shared by node 2 is $3N + 1$ elements, while nodes 3 and 4 transmit a total amount of $3N$ elements each. Alternatively, nodes 2, 3 and 4 can pre-compute $d_2(n) - \theta_{2,2}(n)$, $\theta_{3,2}(n)$ and $\theta_{4,2}(n)$, so that only one scalar is transmitted by each node. One can note that this latter method leads to a reduction in the total information shared by each node of the network. For that reason, we will adopt this latter approach in the rest of the paper. For a K-node network, the i-th element of the network shares $K - 1$ pre-computed parameters $\theta_{i,k}, k \neq i$, and the scalar $\hat{\theta}_{i,i} = d_i(n) - \theta_{i,i}$ resulting in a transmission of K scalars.

Notice that if the network is fully connected, regardless of its cardinality, each node has access to all data available in the network at the same instant. This is not true for topologies which are not fully connected, as depicted in Fig. 2. In this case, we should adopt a different strategy to implement the distributed estimation, as described in the next section.

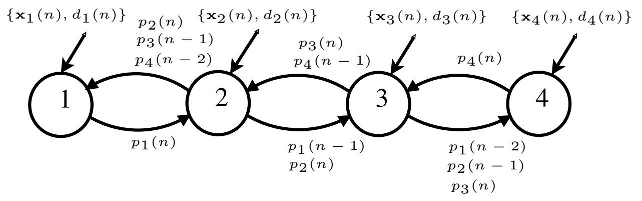
3. MULTICHANNEL DISTRIBUTED NETWORKS (MDNs)

In this section, we propose an algorithm for networks with topologies in which the nodes are not fully connected. In these cases, the nodes must have extra memory to deal with delays between the data received from the different nodes. We



$\{\mathbf{x}_1(n), d_1(n)\}$
$p_1(n)$
$\{\mathbf{x}_2(n), d_2(n)\}$
1
$p_2(n)$
2
$p_3(n)$
$p_1(n)$
$p_3(n)$
$p_1(n)$
$p_4(n)$
$p_2(n)$
$p_4(n)$
$p_2(n)$
3
$p_3(n)$
4
$\{\mathbf{x}_3(n), d_3(n)\}$
$p_4(n)$
$\{\mathbf{x}_4(n), d_4(n)\}$

Fig. 1. 4-channel adaptive filter implemented as a fully-connected DDN.



$\{\mathbf{x}_1(n), d_1(n)\}$
$p_2(n)$
$p_3(n-1)$
$p_4(n-2)$
$\{\mathbf{x}_2(n), d_2(n)\}$
$p_3(n)$
$p_4(n-1)$
$\{\mathbf{x}_3(n), d_3(n)\}$
$p_4(n)$
$\{\mathbf{x}_4(n), d_4(n)\}$
1
2
3
4
$p_1(n)$
$p_1(n-1)$
$p_2(n)$
$p_1(n-2)$
$p_2(n-1)$
$p_3(n)$

Fig. 2. Distributed 4-channel adaptive filter.

describe the problems and solutions through the example of a four-node network implementing the LMS technique.

To simplify our analysis, define the following vectors to describe the information exchanged by the nodes, i.e.

$$\begin{aligned}
p_1(n) &= [(d_1(n)-\theta_{1,1}(n)) \;\; \theta_{1,2}(n) \;\; \theta_{1,3}(n) \;\; \theta_{1,4}(n)]^T \\
p_2(n) &= [\theta_{2,1}(n) \;\; (d_2(n)-\theta_{2,2}(n)) \;\; \theta_{2,3}(n) \;\; \theta_{2,4}(n)]^T \\
p_3(n) &= [\theta_{3,1}(n) \;\; \theta_{3,2}(n) \;\; (d_3(n)-\theta_{3,3}(n)) \;\; \theta_{3,4}(n)]^T \\
p_4(n) &= [\theta_{4,1}(n) \;\; \theta_{4,2}(n) \;\; \theta_{4,3}(n) \;\; (d_4(n)-\theta_{4,4}(n))]^T
\end{aligned} \qquad (7)$$

Consider the network[1] presented in Fig. 2. Node 2 feeds node 1 with data from 3 and 4, since there are no links between nodes 1 and 3 or 1 and 4. At time instant $n-2$, node 3 receives $p_4(n-2)$, which is stored and transmitted to node 2 at time instant $n-1$. At instant $n-1$, node 3 transmits $p_4(n-2)$ and $p_3(n-1)$ to node 2, which stores these terms until instant n, when $p_2(n)$, $p_3(n-1)$ and $p_4(n-2)$ are finally transmitted to node 1. The same procedure is used to re-transmit data from nodes 1 and 2 to node 4, via node 3. Assuming that the estimation performed in nodes 1 and 4 uses only data from the same instant, these nodes must wait until instant n to have all data from instant $n-2$ available. For nodes 2 and 3, one can check that the delay to acquire data is just one tap, so that at instant n, all data from instant $n-1$ is available and can be applied to estimation. Using this approach, nodes 1 and 4 need to store local data from instants $n-1$ and $n-2$, while nodes 2 and 3 store their own data from instant $n-1$ and obtained from their neighbors.

Applying the delayed data re-transmitted by the nodes, the estimated parameters at instant $n+1$ are given by

[1]Other networks can be implemented with a similar approach. This topology is applied to facilitate the explanation of the method.

$$\begin{aligned}
\mathbf{w}_{1,i}(n+1) &= \mathbf{w}_{1,i}(n-2) + \mu e_i(n-2)\mathbf{x}_1(n-2) \\
\mathbf{w}_{2,i}(n+1) &= \mathbf{w}_{2,i}(n-1) + \mu e_i(n-1)\mathbf{x}_2(n-1) \\
\mathbf{w}_{3,i}(n+1) &= \mathbf{w}_{3,i}(n-1) + \mu e_i(n-1)\mathbf{x}_3(n-1) \\
\mathbf{w}_{4,i}(n+1) &= \mathbf{w}_{4,i}(n-2) + \mu e_i(n-2)\mathbf{x}_4(n-2)
\end{aligned} \qquad (8)$$

for $i = 1, 2, 3, 4$. The propagation delay on each node depends upon the number of re-transmissions required to feed an element with the information from the farthest node in the network. Networks with a high number of nodes and less connected topologies can present higher propagation delays. When the delay increases, more data must be stored and transmitted by the nodes, which requires an efficient way to share information. For this purpose, we employ the pre-computation of the $\theta_{j,k}(n)$, so that the data shared by the nodes are as presented in (7). Using (7), nodes 1 and 4 must share 4 terms, while nodes 2 and 3 transmit 20 elements each (4 terms locally computed and 16 terms to be re-transmitted).

Note that for larger networks, the technique may be difficult to implement, since the number of elements that must be stored in each node increases linearly with K, so that the memory requirements can become too large. However, our distributed method can also be applied, if we limit the memory available in each node. The memory can be limited, for instance, in networks for which the correlation between $d_k(n)$ and $\mathbf{x}_j(n)$ is low for some nodes ($k \neq j$). In this case, the contribution of node j can be neglected by node k, with just a small degradation in the mean-square deviation (MSD) performance. Since the data from node j is not applied in the estimation performed in node k, this data is not stored, reducing the memory requirements for node j.

The delays introduced in the network complicate the analysis of the proposed technique, since eq. (8) cannot be studied using traditional tools applied to evaluate the convergence of adaptive filters [12, 13]. In the next section, we show how the system equation of the network of Fig. 2 can be written to allow the numerical computation of a range of values to choose μ, guaranteeing the stability in the mean. The approach is presented for a four-node network, but can be extended to any network.

4. CONVERGENCE IN THE MEAN

Consider the network presented in Fig. 2. To study its convergence in the mean, define

$$\tilde{\mathbf{w}}_{j,i}(n) \triangleq \mathbf{w}^o_{j,i} - \mathbf{w}_{j,i}(n). \qquad (9)$$

Use equations (2) and (9) in (4) to obtain

$$e_i(n) = \sum_{j=1}^{4} \tilde{\mathbf{w}}^T_{j,i}(n)\mathbf{x}_j(n) + v_i(n).$$

Subtract eq. (8) from $\mathbf{w}^o_{j,i}$ ($j = 1, 2, 3, 4$), and define

$$\boldsymbol{\mathcal{R}}_{i,j}(n) = \mathbf{x}_i(n)\mathbf{x}^T_j(n), \;\; \boldsymbol{\mathcal{V}}_{i,j}(n) = \mathbf{x}_i(n)v_j(n) \text{ and}$$

$$\boldsymbol{\Theta}_{k,i}(n) = \sum_{j=1}^{4} \boldsymbol{\mathcal{R}}_{k,j}(n)\tilde{\mathbf{w}}_{j,i}(n) + \boldsymbol{\mathcal{V}}_{k,i}(n)$$

to obtain

$$\begin{aligned}\tilde{\mathbf{w}}_{1,i}(n+1) &= \tilde{\mathbf{w}}_{1,i}(n-2) - \mu\boldsymbol{\Theta}_{1,i}(n-2)\\ \tilde{\mathbf{w}}_{2,i}(n+1) &= \tilde{\mathbf{w}}_{2,i}(n-1) - \mu\boldsymbol{\Theta}_{2,i}(n-1)\\ \tilde{\mathbf{w}}_{3,i}(n+1) &= \tilde{\mathbf{w}}_{3,i}(n-1) - \mu\boldsymbol{\Theta}_{3,i}(n-1)\\ \tilde{\mathbf{w}}_{4,i}(n+1) &= \tilde{\mathbf{w}}_{4,i}(n-2) - \mu\boldsymbol{\Theta}_{4,i}(n-2).\end{aligned} \tag{10}$$

Assume that $v_i(n)$ is zero-mean noise, with variance σ_v^2, independent and identically distributed (i.i.d.). Additionally, assume that $v_i(n)$ is independent from each $\mathbf{x}_j(n)$, such that

$$E\{\mathbf{x}_j(n)v_i(n)\} = E\{\mathbf{x}_j(n)\}E\{v_i(n)\} = 0, \forall_{i,j}.$$

Use the independence approximation [12, 13] usually applied to study the convergence of adaptive filters to write

$$E\{\boldsymbol{\mathcal{R}}_{i,j}(n)\tilde{\mathbf{w}}_{j,k}(n)\} \approx E\{\boldsymbol{\mathcal{R}}_{i,j}(n)\}E\{\tilde{\mathbf{w}}_{j,k}(n)\}.$$

Defining

$$\bar{\boldsymbol{\Theta}}_{k,i}(n) = \sum_{j=1}^{3} \bar{\boldsymbol{\mathcal{R}}}_{k,j}\bar{\mathbf{w}}_{j,i}(n),$$

where $\bar{\mathbf{w}}_{j,i}(n) \triangleq E\{\tilde{\mathbf{w}}_{j,i}(n)\}$ and $\bar{\boldsymbol{\mathcal{R}}}_{k,j} \triangleq E\{\boldsymbol{\mathcal{R}}_{k,j}(n)\}$, one gets

$$\begin{aligned}\bar{\mathbf{w}}_{1,i}(n+1) &= \bar{\mathbf{w}}_{1,i}(n-2) - \mu\bar{\boldsymbol{\Theta}}_{1,i}(n-2)\\ \bar{\mathbf{w}}_{2,i}(n+1) &= \bar{\mathbf{w}}_{2,i}(n-1) - \mu\bar{\boldsymbol{\Theta}}_{2,i}(n-1)\\ \bar{\mathbf{w}}_{3,i}(n+1) &= \bar{\mathbf{w}}_{3,i}(n-1) - \mu\bar{\boldsymbol{\Theta}}_{3,i}(n-1)\\ \bar{\mathbf{w}}_{4,i}(n+1) &= \bar{\mathbf{w}}_{4,i}(n-2) - \mu\bar{\boldsymbol{\Theta}}_{4,i}(n-2).\end{aligned} \tag{11}$$

Define matrix $\bar{\mathbf{W}}(n)$, which contains the vectors $\bar{\mathbf{w}}_{j,k}(n)$, $\forall j,k = 1,2,3,4$, in the positions defined by their indices. Additionally, define the matrix $\mathbf{R}$, in which the block at position j,k is given by $\bar{\boldsymbol{\mathcal{R}}}_{j,k}$, $\forall j,k = 1,2,3,4$. Define

$$\boldsymbol{\Delta} = (\mathbf{I} - \mu\mathbf{R}). \tag{12}$$

Using $\bar{\mathbf{W}}(n)$, $\mathbf{R}$ and $\boldsymbol{\Delta}$, and introducing $\boldsymbol{\mathcal{A}} = \mathbf{0}_{4N}$, $\boldsymbol{\mathcal{B}} = \mathbf{B} \otimes \mathbf{I}_N$ and $\boldsymbol{\mathcal{C}} = \mathbf{C} \otimes \mathbf{I}_N$, where $\mathbf{B} = \text{diag}(0,1,1,0)$ and $\mathbf{C} = \text{diag}(1,0,0,1)$, eq. (11) can be expressed as

$$\bar{\mathbf{W}}(n{+}1) = \boldsymbol{\mathcal{A}}\boldsymbol{\Delta}\bar{\mathbf{W}}(n) + \boldsymbol{\mathcal{B}}\boldsymbol{\Delta}\bar{\mathbf{W}}(n{-}1) + \boldsymbol{\mathcal{C}}\boldsymbol{\Delta}\bar{\mathbf{W}}(n{-}2). \tag{13}$$

Define $\boldsymbol{\omega}(n) = \text{Vec}(\bar{\mathbf{W}}(n))$, where $\text{Vec}(\cdot)$ stacks up the columns of $\bar{\mathbf{W}}(n)$ in $\boldsymbol{\omega}(n)$. Eq. (13) is vectorized to

$$\begin{aligned}\boldsymbol{\omega}(n+1) = [\mathbf{I}_4 \otimes (\boldsymbol{\mathcal{A}}\boldsymbol{\Delta})]\boldsymbol{\omega}(n) + [\mathbf{I}_4 \otimes (\boldsymbol{\mathcal{B}}\boldsymbol{\Delta})]\boldsymbol{\omega}(n-1)\\ + [\mathbf{I}_4 \otimes (\boldsymbol{\mathcal{C}}\boldsymbol{\Delta})]\boldsymbol{\omega}(n-2).\end{aligned} \tag{14}$$

Since eq. (14) is a third-order system of equations, the usual approach to study the dynamical properties of adaptive filters cannot be applied. In order to obtain bounds to select a range of step sizes which guarantee the convergence in the mean, one can write

$$\tilde{\boldsymbol{\omega}}(n) = \mathbf{U}\tilde{\boldsymbol{\omega}}(n-1), \tag{15}$$

where

$$\mathbf{U} = \begin{bmatrix} [\mathbf{I}_4 \otimes (\boldsymbol{\mathcal{A}}\boldsymbol{\Delta})] & [\mathbf{I}_4 \otimes (\boldsymbol{\mathcal{B}}\boldsymbol{\Delta})] & [\mathbf{I}_4 \otimes (\boldsymbol{\mathcal{C}}\boldsymbol{\Delta})] \\ \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix},$$

and study the convergence in the mean accessing the eigenvalues of $\mathbf{U}$. Since the eigenvalue decomposition of $\mathbf{U}$ is not trivial, one can write this matrix using values obtained from a given problem and numerically compute a range of values for μ for which the absolute value of each eigenvalue is less than 1, guaranteeing the stability. A similar approach can be applied to write $\mathbf{U}$ for an arbitrary number of nodes.

5. SIMULATIONS

In this section, we compare the performance of the proposed technique and the multichannel LMS algorithm, implemented as a fully-connected DDN. For this purpose, we consider the 7-Ch-LMS algorithm and our approach to implement the distributed 7-Ch-LMS in the network presented in Fig. 3. The proposed technique is implemented in two different ways. In the first approach, the data of each element is shared with all nodes of the network, so that the maximum propagation delay is 5 taps. In the second approach, we assume that the correlation between $d_k(n)$ and $\mathbf{x}_j(n)$ decreases as the distance between nodes k and j enlarges, so that the contribution of far nodes to the estimated coefficients is low and can be disregarded. In this case, the amount of data transmitted and stored by the nodes is reduced, allowing us to limit the available memory in each element. In our simulation, the reduction in correlation between $d_k(n)$ and $\mathbf{x}_j(n)$ is obtained by choosing the $\mathbf{w}_{k,j}^o$ such that $||\mathbf{w}_{k,j}^o||_2 \propto c(k,j)$, with $c(k,j) = 1$ if $k = j$, and $c(k,j) = 1/4^{m+1}$ if m nodes are required to re-transmit data from node j to node k. It is assumed that if $c(k,j) < 1/64$, then the contribution of node j can be excluded from the estimation performed by node k. For the network presented in Fig. 3, this implies in a maximum delay of 2 taps. $v_k(n)$ is i.i.d. Gaussian noise, with zero mean and variance 10^{-3}, independent from $\mathbf{x}_k(n)$, $k = 1,2,\ldots,K$. The figure of merit used in the comparison is the MSD, which is computed for each node and then applied to obtain the network MSD, given by

$$\text{MSD}_{\text{Network}} = \frac{1}{K}\sum_{k=1}^{K}\text{MSD}_k,$$

where MSD_k is computed for the k-th node. The input data $\mathbf{x}_k(n)$, $k = 1,2,\ldots,K$, is a 20-entry tap delay-line, and each element of $\mathbf{x}_k(n)$ is zero-mean, Gaussian, i.i.d. and with unitary variance. At the beginning of the simulation, we randomly define the set of optimum parameters $\mathbf{W}^o$, which is normalized so that the 2-norm of the matrix $\mathbf{W}^o$ is unitary ($||\mathbf{W}^o||_2 = 1$). These values for the entries of $\mathbf{W}^o$ are kept the same for all iterations. The step size used for each algorithm is indicated in Fig. 4, where we also present the curve obtained when there is no collaboration among the nodes, so that seven LMS filters operate independently from each other. The curves are obtained by average of 200 realizations.

Note from Fig. 4 that the proposed technique achieves almost the same MSD performance of the 7-Ch-LMS algo-

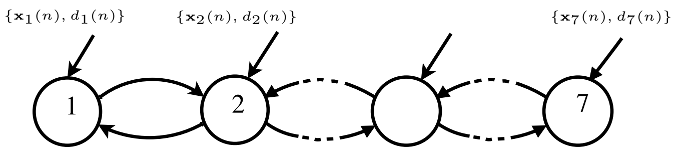

$\{\mathbf{x}_1(n), d_1(n)\}$
$\{\mathbf{x}_2(n), d_2(n)\}$
$\{\mathbf{x}_7(n), d_7(n)\}$
1
2
7

Fig. 3. Distributed 7-channel adaptive filter.

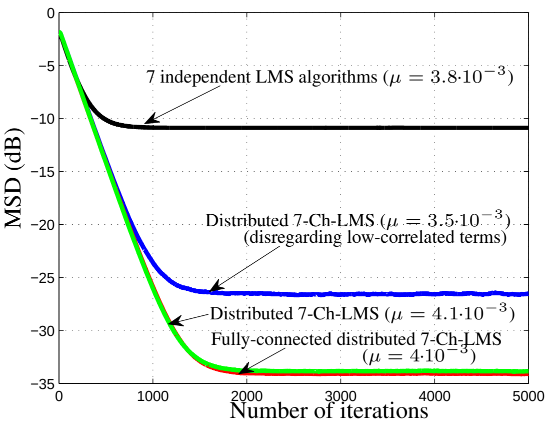


Fig. 4. MSD comparison between the 7-Ch-LMS and the distributed 7-Ch-LMS algorithms. Mean of 200 realizations.

rithm, if every node in our approach has access to the delayed data from the other elements of the network. If our method dismisses data from far nodes, a performance degradation occurs, but the algorithm still outperforms the LMS algorithms operating separately.

6. FINAL REMARKS

In this paper, we showed that a mutichannel adaptive filter can be interpreted as a fully-connected, delayless distributed network, in which each channel corresponds to a node. More general distributed networks, in which the nodes are not fully connected, were also considered. In this case, we showed that these networks can still be implemented with multichannel adaptive filters through the introduction of memory in the nodes. For larger networks, the amount of information stored and transmitted by each node can become impractical with the increase of K. In order to mitigate this problem, we limit the available memory in each node. If the correlation between $d_k(n)$ and $\mathbf{x}_j(n)$ is low, we can neglect the contribution of node j to the estimation performed in node k, reducing the data that must be stored in this node. A small degradation is introduced in the MSD performance, but the algorithm still outperforms adaptive filters operating without collaboration.

In future works we intend to develop models for the mean-square performance of MDNs.

7. REFERENCES

[1] J. Chou, D. Petrovic, and K. Ramachandran, "A distributed and adaptive signal processing approach to reducing energy consumption in sensor networks," in *Twenty-Second Annual Joint Conference of the IEEE Computer and Communications*, March 2003, vol. 2, pp. 1054–1062 vol.2.

[2] R. Olfati-Saber and R.M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. Automat. Contr.*, vol. 49, no. 9, pp. 1520–1533, Sept 2004.

[3] S.-Y. Tu and A.H. Sayed, "Tracking behavior of mobile adaptive networks," in *The Forty Fourth Asilomar Conference on Signals, Systems and Computers*, Nov 2010, pp. 698–702.

[4] F.S. Cattivelli and A.H. Sayed, "Modeling bird flight formations using diffusion adaptation," *IEEE Trans. Signal Processing*, vol. 59, no. 5, pp. 2038–2051, May 2011.

[5] J. Li and A.H. Sayed, "Modeling bee swarming behavior through diffusion adaptation with asymmetric information sharing," in *EURASIP Journal on Advances in Signal Processing, 2012:18*, 2012.

[6] C.G. Lopes and A.H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Processing*, vol. 55, no. 8, pp. 4064 –4077, aug. 2007.

[7] S. Kar and J.M.F. Moura, "Consensus + innovations distributed inference over networks: cooperation and sensing in networked systems," *IEEE Signal Processing Mag.*, vol. 30, no. 3, pp. 99–109, May 2013.

[8] C.G. Lopes and A.H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," *IEEE Trans. Signal Processing*, vol. 56, no. 7, pp. 3122 –3136, july 2008.

[9] A.H. Sayed, "Diffusion adaptation over networks," in *Academic Press Library in Signal Processing:*, Rama Chellappa and Sergios Theodoridis, Eds., vol. 3, Signal Processing Theory and Machine Learning, pp. 323–454. Chennai: Academic Press, 2014.

[10] A.H. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z.J. Towfic, "Diffusion strategies for adaptation and learning over networks: an examination of distributed strategies and network behavior," *IEEE Signal Processing Mag.*, vol. 30, no. 3, pp. 155–171, May 2013.

[11] R.A. Horn and C.R. Johnson, *Topics in Matrix Analysis*, Cambridge University Press, Cambridge, MA, 1991.

[12] A.H. Sayed, *Fundamentals of adaptive filtering*, John Wiley & Sons, 2003.

[13] V.H. Nascimento and M.T.M. Silva, "Adaptive filters," in *Academic Press Library in Signal Processing:*, Rama Chellappa and Sergios Theodoridis, Eds., vol. 1, Signal Processing Theory and Machine Learning, pp. 619—761. Chennai: Academic Press, 2014.