Distributed estimation in diffusion networks using affine least-squares combiners[☆]

Jesus Fernandez-Bes[*,a,1], Luis A. Azpicueta-Ruiz[a,1], Jerónimo Arenas-García[a,1], Magno T. M. Silva[b,2]

[a] *Dept. Signal Theory and Communications, Universidad Carlos III de Madrid, Leganés, 28911, Spain*
[b] *Dept. Electronic Systems Engineering, Escola Politécnica, Universidade de São Paulo, São Paulo, 05508-010, Brazil*

Abstract

We propose a diffusion scheme for adaptive networks, where each node obtains an estimate of a common unknown parameter vector by combining a local estimate with the combined estimates received from neighboring nodes. The combination weights are adapted in order to minimize the mean-square error of the network employing a local least-squares (LS) cost function. This adaptive diffusion network with LS combiners (ADN-LS) is analyzed, deriving expressions for its network mean-square deviation that characterize the convergence and steady-state performance of the algorithm. Experiments carried out in stationary and tracking scenarios show that our proposal outperforms an state-of-art scheme for adapting the weights of diffusion networks (ACW algorithm from [10]), both during convergence and in tracking situations. Despite its good convergence behavior, our proposal may present a slightly worse steady-state performance in stationary or slowly-changing scenarios with respect to ACW due to the error inherent to the least-squares adaptation with sliding window. Therefore, to take advantage of these different behaviors, we also propose a hybrid scheme based on a convex combination of the ADN-LS and ACW algorithms.

Key words: Adaptive filters, adaptive diffusion networks, distributed estimation, combination of filters, least-squares, statistical analysis.

[☆] Some preliminary parts of this work appeared as a conference paper [1].

[*] Corresponding author

[1] The work of Fernandez-Bes, Azpicueta-Ruiz, and Arenas-García was partly supported by MINECO projects TEC2011-22480 and PRI-PIBIN-2011-1266. The work of Fernandez-Bes was also partly supported by Spanish MECD FPU program. The work of Azpicueta-Ruiz was also partly supported by FAPESP under Grant 2013/18041-5.

[2] The work of Silva was partly supported by CNPq under Grant 302423/2011-7 and FAPESP under Grant 2012/24835-1.

Email addresses: {jesusfbes,lazpicueta,jarenas}@tsc.uc3m.es, magno@lps.usp.br

1. Introduction

Over the last years, adaptive networks have gained considerable attention as an efficient solution to estimate certain parameters of interest using the information from data collected at nodes distributed over a region (see e.g., [2]-[12] and their references). Many applications reach an improved behavior thanks to the use of data measured in different localizations, e.g., target localization and tracking, considering either a static or a mobile network [12, 13], environment monitoring [2], and spectrum sensing in mobile networks, where secondary users can estimate the power spectrum transmitted by all the primary users to adaptively find unused frequency bands [12, 14].

In these applications, networks must track the variations in the data statistics, which justifies the need for adaptiveness [15]. It should be noted that the tracking ability of adaptive networks constitutes one of their advantages with respect to other distributed estimation schemes, such as consensus networks [12]. Additionally, some networks operate under computational, communication or energy constraints, which makes the signal processing problem even more challenging [16, 17].

Different distributed processing strategies have been designed to build an estimate of some parameters of interest by exchanging information among the network nodes. In particular, in diffusion networks nodes diffuse their estimates to the network, so that each node can combine its own estimation with those received from neighboring nodes. The diffusion strategy is typically performed in two stages: adaptation and combination. The order in which these stages are performed leads to two possibilities: adapt-then-combine (ATC) and combine-then-adapt (CTA) [3, 5]. In ATC, each node updates its local estimate using the combined estimate from the previous iteration. Then, the local estimates of the nodes belonging to the neighborhood are mixed to update the combined estimate. On the other hand, in CTA the combined estimate is firstly updated by mixing the estimates received from the neighboring nodes, and then used to update the local estimate. In both diffusion schemes, the combination weights play an essential role in the overall performance of the network. For instance, diffusion least-mean-squares (LMS) strategies for distributed estimation can perform similarly to classical centralized solutions when the weights used to combine the different estimates are optimally adjusted [9]-[12].

Most papers on diffusion networks assume fixed combination weights, whose values are computed based on the network topology only (see e.g., [3, 5]). However, these static combination rules do not take into account diversity among nodes, or different signal-to-noise ratio (SNR) conditions, resulting in suboptimal performance when the SNR varies across the network. For this reason, some schemes that implement adaptive combination weights have been recently proposed [9]-[12],[18]. Although these adaptive solutions improve the performance of the networks when compared to static combiners, the learned combination parameters may still be suboptimal during convergence or when tracking time-varying solutions, especially when different step sizes are used across the network nodes. This is due to the fact that some of the approximations used in the derivation of these adaptive rules hold mainly in steady state.

In this paper, we propose a new approach to adapt the combiners of diffusion networks which can improve network performance, especially during convergence or in tracking situations. Unlike previous schemes, the adaptive combiners that we propose here are based on the direct minimization of a least-squares cost function that considers the data available at each node. For this, we exploit some recent advances in the literature regarding combination of adaptive filters.

In an adaptive combination of filters, different schemes can be considered to mix the outputs of the component filters, including convex [19]-[23] and affine [24]-[26] combinations. In [26], a least-squares (LS) scheme was proposed to adapt an affine combination of multiple adaptive filters, providing an adequate behavior in stationary and nonstationary environments. It is important to notice that in a combination of multiple adaptive filters, all the filters (or nodes) receive the same input vector, there is a common desired signal, and the component filters do not exchange information in general[3], so the extension of these combination schemes to adaptive networks is not straightforward.

As in our previous work [1], we utilize here LS-based affine combinations [26] to implement adaptive combiners in diffusion networks. This scheme, named Adaptive Diffusion Network with Least-Squares combiners (ADN-LS), presents the following

[3]Indeed, [19] proposed a scheme for coefficient transfer between the filters. In this scheme, the convergence of the slow filter can be accelerated when an abrupt change appears by transferring a part of the fast filter to the slow one. Therefore, a combination of adaptive filters also allows exchange of information among their components.

characteristics:

- It follows an ATC strategy (although our results could also be straightforwardly extended to CTA). However, an important difference from standard ATC is that each node preserves a pure local estimation.

- Each node combines its local estimate with the combined estimates received from neighboring nodes at the previous iteration. Combination weights are adapted to minimize a local LS cost function employing a sliding window [26].

In this paper, we extend our workshop paper [1] in different ways. Specifically, the main contributions of this paper are:

1. We provide a statistical analysis to model the transient and steady-state performance of ADN-LS. We arrive at analytical expressions for the mean-square deviation (MSD) and for the optimal combiners. This analysis is essential to understand the behavior of ADN-LS with optimal combiners in the transient and in the steady state, and allows us to obtain the performance limits of the proposed algorithm. To the best of our knowledge, there is no transient analysis of a diffusion scheme with adaptive combiners currently available in the literature.
2. A new hybrid scheme based on a convex combination of ADN-LS with the scheme of [10]. As we will see, ADN-LS outperforms the state-of-the-art scheme of [10], denominated as Adaptive Combination Weights (ACW) in [12], during convergence and in tracking situations. However, the steady-state performance of ADN-LS can be slightly worse than that of ACW in certain cases. In this sense, the hybrid scheme takes advantage of these somehow complementary properties.
3. A discussion on the computational and communication costs of the proposed schemes.
4. Finally, we provide detailed simulation work to illustrate the performance of our schemes. This experimental evaluation extends the results in [1] with respect to the considered algorithms and simulation scenarios. Additionally, we also study the influence of the sliding window necessary for the adaptation of ADN-LS combiners in the performance of the algorithm.

The rest of the paper is organized as follows. The next section presents the principles of adaptive diffusion networks and describes our proposal for implementing adaptive

combiners that minimize an LS criterion. Then, Section 3 provides a statistical analysis of ADN-LS with optimal combiners. The hybrid scheme combining ADN-LS and ACW rules is presented in Section 4, together with an evaluation of the computational and communication costs of our schemes. Section 5 studies the stationary and tracking capabilities of our proposal, both in the illustrative case of a simple network, and for a more involved diffusion network with 15 nodes. We finish the paper by presenting our main conclusions and some possibilities for future research.

Table 1 summarizes the notation that will be used in the paper.

Table 1: Summary of the notation used in the paper.

N	Number of nodes in the network
$\mathcal{N}_k$	Neighborhood of node k, including itself
N_k	Cardinality of $\mathcal{N}_k$
$\bar{\mathcal{N}}_k$	Neighborhood of node k, excluding itself
$\bar{N}_k$	Cardinality of $\bar{\mathcal{N}}_k$
$\bar{\mathbf{b}}_k$	Vector with the indexes of all nodes belonging to $\bar{\mathcal{N}}_k$
$\bar{b}_k^{(m)}$	Index of the m^{th} node connected to node k
$\mathbf{w}_{\text{o}}(n)$	Unknown time-varying parameter vector
$\boldsymbol{\psi}_k(n)$	Local estimate of $\mathbf{w}_{\text{o}}(n)$ (based only on local data at node k)
$\mathbf{w}_k(n)$	Combined estimate of $\mathbf{w}_{\text{o}}(n)$ at node k
$\{d_k(n), \mathbf{u}_k(n)\}$	Local desired value and regression vector at node k
$v_k(n)$	Local noise at node k (with mean zero and variance σ_k^2)
$c_{\ell k}(n)$	Combination weight assigned by node k to the estimate of node $\ell \in \mathcal{N}_k$
$\mathbf{c}_k(n)$	Vector with all weights assigned by node k to its neighbors
$\bar{\mathbf{c}}_k(n)$	Vector with the same entries of $\mathbf{c}_k(n)$, excluding $c_{kk}(n)$
$y_k(n)$	Local output of node k
$e_k(n)$	Local error of node k
$\check{y}_k(n)$	Output of node k using combined estimates $\mathbf{w}_k(n)$
$\check{e}_k(n)$	Error of node k using combined estimates $\mathbf{w}_k(n)$

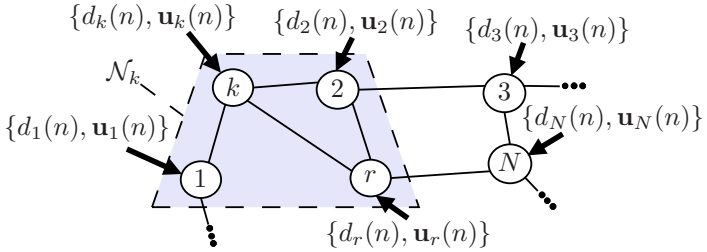


Figure 1: Diffusion network with N nodes: At time n, every node k takes a measurement $\{d_k(n), \mathbf{u}_k(n)\}$. The neighborhood of node k in this network is $\mathcal{N}_k = \{1, 2, r, k\}$, and its cardinality is denoted as $N_k = 4$.

2. Diffusion networks with least-squares combiners

In this section, the ATC strategy based on the preservation of the local estimates at each node is described. Then, we present the LS adaptation for updating the adaptive combiners.

2.1. Diffusion networks principles

In this paper, we focus on estimation tasks which are jointly carried out by multiple nodes in a diffusion network. Fig. 1 shows a network composed by N nodes distributed over some region. The set of nodes connected to node k (including k itself) is called the neighborhood of node k and is denoted by $\mathcal{N}_k$, with cardinality N_k. We define $\bar{\mathcal{N}}_k$ as the neighborhood of node k, excluding such node, and its cardinality is denoted as $\bar{N}_k = N_k - 1$. For instance, for node k of Fig. 1, $\mathcal{N}_k = \{1, 2, r, k\}$, $N_k = 4$, $\bar{\mathcal{N}}_k = \{1, 2, r\}$, and $\bar{N}_k = 3$.

At each time instant n, each node takes a measurement $\{d_k(n), \mathbf{u}_k(n)\}$, where $d_k(n)$ represents a desired signal and $\mathbf{u}_k(n)$ is a length-M input regressor column vector, related through a linear regression model [15]

$$d_k(n) = \mathbf{u}_k^T(n)\mathbf{w}_{\mathrm{o}}(n-1) + v_k(n). \tag{1}$$

Here, $v_k(n)$ is a zero-mean noise with variance σ_k^2, while $\mathbf{w}_{\mathrm{o}}(n)$ is a length-M column parameter vector. Note that $\mathbf{w}_{\mathrm{o}}(n)$ is assumed to be common to the regression models of all nodes, and the goal of the network is precisely the estimation of this unknown parameter vector.

Similarly to other diffusion strategies, we assume that the nodes obtain improved estimates by iterating adaptation and combination steps. However, unlike other schemes

that modify the adaptation of nodes using the combined estimates, we consider that the nodes in our network keep and update a purely local estimate of the parameter vector $\mathbf{w}_{\mathrm{o}}(n)$, i.e., based just on observations available to them.

The local estimates $\boldsymbol{\psi}_k(n)$, $k = 1, 2, \ldots, N$, are updated using the normalized least-mean-squares (NLMS) algorithm [15, 27], i.e.,

$$\boldsymbol{\psi}_k(n) = \boldsymbol{\psi}_k(n-1) + \frac{\mu_k}{\delta + \|\mathbf{u}_k(n)\|^2} e_k(n)\mathbf{u}_k(n), \tag{2}$$

where $e_k(n) = d_k(n) - \mathbf{u}_k^T(n)\boldsymbol{\psi}_k(n-1) \triangleq d_k(n) - y_k(n)$ is the local error of node k, with $y_k(n) = \mathbf{u}_k^T(n)\boldsymbol{\psi}_k(n-1)$ being the predicted value of the desired signal at node k and time n using local information only. Parameter μ_k is a step size that controls the velocity of adaptation of the algorithm, imposing a tradeoff between speed of convergence (faster for large values of μ_k) and steady-state error (lower for small values of μ_k). Since NLMS normalizes its adaptation by the energy of the input signal vector ($\|\mathbf{u}_k(n)\|^2$) selection of the step size is easy, with values in the range $0 < \mu_k < 2$ guaranteeing the convergence and stability of the algorithm. Additionally, a small positive constant δ is included as a regularization parameter in order to avoid a very large *normalized* step size $\frac{\mu_k}{\delta + \|\mathbf{u}_k(n)\|^2}$ when $\|\mathbf{u}_k(n)\|^2$ becomes close to zero.

In addition to local estimates, all nodes build a combined estimate $\mathbf{w}_k(n)$ using the information received from their neighboring nodes. To be more specific, each node k combines its local estimate $\boldsymbol{\psi}_k(n)$ with the combined estimates received from its neighbors at iteration $n-1$, i.e.,

$$\mathbf{w}_k(n) = c_{kk}(n)\boldsymbol{\psi}_k(n) + \sum_{\ell \in \bar{\mathcal{N}}_k} c_{\ell k}(n)\mathbf{w}_\ell(n-1), \tag{3}$$

where $c_{\ell k}(n)$, for $k = 1, 2, \ldots, N$ and $\ell \in \mathcal{N}_k$, are the weights assigned by each node k to the different estimates (local and received from its neighbors) combined in (3). In this paper, we assume that all nodes implement affine combinations, i.e., $\sum_{\ell \in \mathcal{N}_k} c_{\ell k}(n) = 1$. It should be evident that an adequate selection of the combination weights can potentially provide an improved estimate of $\mathbf{w}_{\mathrm{o}}(n)$ with respect to $\boldsymbol{\psi}_k(n)$.

Overall, the strategy we have reviewed operates in two stages, local adaptation and combination, and is thus similar to *adapt-then-combine* diffusion extensively studied in the literature (see, e.g., [3, 5, 9] and their references). However, as we have already mentioned, an important difference exists, since in our case combined weights $\mathbf{w}_k(n)$ are not fed back to the local estimate, i.e., replacing $\boldsymbol{\psi}_k(n-1)$ in the right-hand side

term of (2). As we will see later, this is important when using nodes that adapt with different speeds, since, e.g., otherwise feeding weights coming from slow nodes could slow down the convergence of fast nodes [28].

2.2. Least-squares update of the combination weights

Several strategies have already been proposed to select combination weights in diffusion networks. In this section, we present a scheme for implementing adaptive combiners that automatically adjust their values to the current network conditions, e.g., SNR or misadjustment at each node. In this way, we can get a more robust behavior when the statistics of the scenario in which the network operates are unknown or vary with time.

For notation convenience, we define $\mathbf{c}_k(n)$ and $\bar{\mathbf{c}}_k(n)$ as column vectors of lengths N_k and $\bar{N}_k$, containing the combination weights assigned to each of the nodes connected to node k, including $c_{kk}(n)$ and excluding this weight, respectively. We define also the column vector $\bar{\mathbf{b}}_k$ of length $\bar{N}_k$ containing the indexes of all nodes belonging to $\bar{\mathcal{N}}_k$. Thus, $\bar{b}_k^{(f)},\ f = 1, \ldots, \bar{N}_k$, stands for the f^{th} component of $\bar{\mathbf{b}}_k$ and corresponds to the index of the f^{th} node connected to node k. For instance, for node k of Fig. 1, we have $\mathbf{c}_k(n) = [c_{1k}(n)\, c_{2k}(n)\, c_{rk}(n)\, c_{kk}(n)]^T$, $\bar{\mathbf{c}}_k(n) = [c_{1k}(n)\, c_{2k}(n)\, c_{rk}(n)]^T$, $\bar{\mathbf{b}}_k = [\,1\ \ 2\ \ r\,]^T$, and $\bar{b}_k^{(3)} = r$. Following recent works in diffusion networks [9, 12], we assume that the elements of $\mathbf{c}_k(n)$, $k = 1, 2, \ldots, N$, sum up to one, i.e., $\mathbf{1}^T\mathbf{c}_k(n) = \mathbf{1}^T\bar{\mathbf{c}}_k(n) + c_{kk}(n) = 1$, where $\mathbf{1}$ stands for a column vector of matching length with all its elements equal to one. Then, we simply adapt $\bar{\mathbf{c}}_k(n)$ and consider

$$c_{kk}(n) = 1 - \sum_{\ell \in \bar{\mathcal{N}}_k} c_{\ell k}(n). \tag{4}$$

As it is usually the case, we attempt to minimize the network mean-square-error (MSE), which is defined as

$$\text{MSE}(n) = \frac{1}{N}\sum_{k=1}^{N} \text{MSE}_k(n) = \frac{1}{N}\sum_{k=1}^{N} \text{E}\{\check{e}_k^2(n)\}, \tag{5}$$

where $\text{E}\{\cdot\}$ represents the mathematical expectation operator and $\check{e}_k(n)$ stands for the combined error at node k, i.e.,

$$\check{e}_k(n) = d_k(n) - \mathbf{u}_k^T(n)\mathbf{w}_k(n-1) \triangleq d_k(n) - \check{y}_k(n). \tag{6}$$

Hereafter, we will use the inverted hat symbol to denote the output or the error of the nodes that are obtained using combined estimations.

Since only the k^{th} term in the sum (5) depends on $\bar{\mathbf{c}}_k(n)$, adaptation of these combination weights can be done locally at node k with the goal of minimizing the local MSE:

$$\bar{\mathbf{c}}_{k,\text{o}}(n) = \arg\min_{\bar{\mathbf{c}}_k(n)} \text{E}\{\check{e}_k^2(n)\}, \tag{7}$$

where $\bar{\mathbf{c}}_{k,\text{o}}(n)$ are the optimal combination weights at node k and iteration n. However, since nodes only have access to instantaneous measurements, $\{d_k(n), \mathbf{u}_k(n)\}$, the above mathematical expectation cannot be computed. For this reason, it is necessary to employ some approximations relying only on the available data.

In this paper, we propose to carry out the adaptation of $\bar{\mathbf{c}}_k(n)$ to minimize the weighted-sum-of-squares function given by [26, 29]

$$J_k(n) = \sum_{i=1}^{n} \lambda(n,i)\check{e}_k^2(n,i), \tag{8}$$

where $\lambda(n,i)$ is a temporal weighting window,

$$\check{e}_k(n,i) = \; d_k(i) - \check{y}_k(n,i) \tag{9}$$

is the combined error of node k for the i^{th} regressor using the combination weights at time n, and

$$\begin{aligned}\check{y}_k(n,i) &= \sum_{\ell\in\bar{\mathcal{N}}_k} c_{\ell k}(n)\check{y}_{k,\ell}(i) + \left[1 - \sum_{\ell\in\bar{\mathcal{N}}_k} c_{\ell k}(n)\right] y_k(i) \\ &= y_k(i) + \sum_{\ell\in\bar{\mathcal{N}}_k} c_{\ell k}(n)\left[\check{y}_{k,\ell}(i) - y_k(i)\right]\end{aligned} \tag{10}$$

represents the combined output of node k at time i using the combination weights at time n. Note that (4) was used to obtain an expression that does not depend on $c_{kk}(n)$, so that optimization with respect to $c_{\ell k}(n)$ can be done without constraints. Moreover,

$$\check{y}_{k,\ell}(n) = \mathbf{u}_k^T(n)\mathbf{w}_\ell(n-1), \;\; \ell \in \bar{\mathcal{N}}_k \tag{11}$$

is the output of node k filtering its own regression vector $\mathbf{u}_k(n)$ with the estimate provided by neighbor ℓ.

Inserting (10) into (9), we obtain

$$\check{e}_k(n,i) = e_k(i) + \sum_{\ell\in\bar{\mathcal{N}}_k} c_{\ell k}(n)\left[y_k(i) - \check{y}_{k,\ell}(i)\right]. \tag{12}$$

Taking now the derivatives of (8) with respect to each combination weight $c_{mk}(n)$, with $m \in \bar{\mathcal{N}}_k$, we obtain

$$\frac{\partial J_k(n)}{\partial c_{mk}(n)} = 2\sum_{i=1}^{n} \lambda(n,i)\check{e}_k(n,i)\left[y_k(i) - \check{y}_{k,m}(i)\right]. \tag{13}$$

Replacing (12) in (13) and setting the result to zero, after some algebraic manipulations, we obtain

$$\sum_{i=1}^{n}\sum_{\ell\in\bar{\mathcal{N}}_k} \lambda(n,i)c_{\ell k}(n)\tilde{y}_{k,\ell}(i)\tilde{y}_{k,m}(i) = \sum_{i=1}^{n}\lambda(n,i)e_k(i)\tilde{y}_{k,m}(i), \tag{14}$$

where $\tilde{y}_{k,m}(n) \triangleq \check{y}_{k,m}(n) - y_k(n)$, $m \in \bar{\mathcal{N}}_k$.

Since for each node k we have a system with $\bar{N}_k$ equations of the form (14), we introduce the usual matrix notation, i.e.,

$$\mathbf{P}_k(n)\bar{\mathbf{c}}_k(n) = \mathbf{z}_k(n), \tag{15}$$

where $\mathbf{P}_k(n)$ is a square symmetric matrix of size $\bar{N}_k$ with components

$$[\mathbf{P}_k(n)]_{f,g} = \sum_{i=1}^{n}\lambda(n,i)\tilde{y}_{k,\bar{b}_k^{(f)}}(i)\tilde{y}_{k,\bar{b}_k^{(g)}}(i), \tag{16}$$

with $f, g = 1, 2, \ldots, \bar{N}_k$, and $\mathbf{z}_k(n)$ is a column vector of length $\bar{N}_k$, whose f^{th} element is given by

$$z_k^{(f)}(n) = \sum_{i=1}^{n}\lambda(n,i)e_k(i)\tilde{y}_{k,\bar{b}_k^{(f)}}(i), \tag{17}$$

for $f = 1, 2, \ldots, \bar{N}_k$.

Thus, the solution of the problem is obtained from (15) as

$$\bar{\mathbf{c}}_k(n) = \mathbf{P}_k^{-1}(n)\mathbf{z}_k(n). \tag{18}$$

Similarly to the case of combination of multiple filters [26], $\mathbf{P}_k(n)$ can be interpreted as an estimation of $\mathbf{P}_{k,\mathrm{o}}(n) \triangleq \mathrm{E}\{\tilde{\mathbf{y}}_k(n)\tilde{\mathbf{y}}_k^T(n)\}$, the autocorrelation matrix of vector $\tilde{\mathbf{y}}_k(n) = [\tilde{y}_{k,\bar{b}_k^{(1)}}(n) \quad \tilde{y}_{k,\bar{b}_k^{(2)}}(n) \quad \cdots \quad \tilde{y}_{k,\bar{b}_k^{(\bar{N}_k)}}(n)]^T$, while $\mathbf{z}_k(n)$ is an estimation of $\mathbf{z}_{k,\mathrm{o}}(n) \triangleq \mathrm{E}\{e_k(n)\tilde{\mathbf{y}}_k(n)\}$, the cross-correlation vector between $\tilde{\mathbf{y}}_k(n)$ and $e_k(n)$.

Using the same arguments of [25] and [26], we assume that $\lambda(n,i)$ is a rectangular window with length L, i.e.,

$$\lambda(n,i) = \begin{cases} 1, & n-i < L \\ 0, & n-i \geq L. \end{cases} \tag{19}$$

The choice of the window length L is generally not straightforward, and is subject to a well-known trade-off between convergence capabilities (faster for small L) and steady-state performance (better for large L). First of all, $L \gg \bar{N}_k$ to avoid as much as possible the ill-conditioning of matrix $\mathbf{P}_k(n)$. Secondly, we should remark that the estimation of the optimal combination weights is itself a time-varying problem and according to the literature [29]-[31], there is an optimum window that depends on the particular filtering scenario. As a consequence of this, we will select a not very large window (for instance, we select $L = 100$ in our simulations) to guarantee good tracking capability at the cost of slightly increasing the steady-state error in certain situations. An additional advantage of using a not too large window length is that the increased variance in the estimation of $\mathbf{P}_{k,\mathrm{o}}(n)$ can act as a sort or regularizer, and have a stabilization effect in the inversion carried out in (18).

3. Statistical analysis of the proposed scheme

In this section we present a theoretical model for the performance of an adaptive network using the proposed diffusion scheme and LS-based adaptation of the combination weights. Before that, the next subsection describes some assumptions and definitions that will be necessary for the analysis.

3.1. Data model and definitions

We start by introducing some simplifying assumptions:

A1- We assume a random-walk nonstationary environment, where the unknown parameter vector $\mathbf{w}_{\mathrm{o}}$ varies as

$$\mathbf{w}_{\mathrm{o}}(n) = \mathbf{w}_{\mathrm{o}}(n-1) + \mathbf{q}(n), \tag{20}$$

where $\mathbf{q}(n)$ are zero-mean independent and identically distributed (i.i.d.) vectors with positive-definite autocorrelation matrix $\mathbf{Q} = \mathrm{E}\{\mathbf{q}(n)\mathbf{q}^T(n)\}$, independent of the initial conditions $\boldsymbol{\psi}_k(0)$, $\mathbf{w}_k(0)$ and of $\{\mathbf{u}_k(n'), v_k(n')\}$ for all k and n'. Although this model implies that the covariance matrix of $\mathbf{w}_{\mathrm{o}}(n)$ grows to infinity as $n \to \infty$, it has been commonly used in the literature to keep the analysis of adaptive systems simpler [15, 32].

A2- Input regressors are zero-mean and have covariance matrix $\mathbf{R}_{kk} = \mathrm{E}\{\mathbf{u}_k(n)\mathbf{u}_k^T(n)\}$. Furthermore, regressors are spatially independent, i.e.,

$$\mathbf{R}_{k\ell} = \mathrm{E}\{\mathbf{u}_k(n)\mathbf{u}_\ell^T(n)\} = \mathbf{0}, \quad k \neq \ell,$$

where $\mathbf{0}$ is a vector or matrix with appropriate dimensions and all elements equal to zero. This assumption is widely employed in the analysis of diffusion algorithms and is realistic in many practical applications [12].

Furthermore, the noise process $\{v_k(n)\}$ is assumed to be temporally white and spatially independent, i.e.

$$\begin{aligned} &\mathrm{E}\{v_k(n)v_k(n')\} = 0, \ \text{ for all } \ n \neq n' \qquad \text{and} \\ &\mathrm{E}\{v_k(n)v_\ell(n')\} = 0, \ \text{ for all } \ n, n' \ \text{ whenever } \ k \neq \ell. \end{aligned}$$

Additionally, the noise is assumed to be independent (not only uncorrelated) of the regression data $\mathbf{u}_\ell(n')$, so that $\mathrm{E}\{v_k(n)\mathbf{u}_\ell(n')\} = \mathbf{0}$, for all $k, \ell, n,$ and n'. Under Assumption **A2**, $\boldsymbol{\psi}_k(n-1)$ is independent of $v_\ell(n)$ and $\mathbf{u}_\ell(n)$, for all k, and ℓ. This condition matches well with simulation results for sufficiently small step sizes, even when the independence assumption does not hold [12]. A similar condition can be observed in the behavior of stand-alone adaptive filters [15, 27, 33, 34] and is widely used in the majority of analyses of diffusion schemes [9]-[12].

A3- The adaptation of the combination weights $c_{\ell k}(n)$ is slow when compared to the adaptation of the local and combined estimates. Therefore, the correlation between combination parameters and local and combined estimates can be disregarded. This assumption follows from observations: simulations show that the combination weights converge slowly compared to variations in the input regressor $\mathbf{u}_k(n)$ and thus to variations on the local and combined estimates.

A4- Finally, we also assume that $\boldsymbol{\psi}_k(n) \approx \boldsymbol{\psi}_k(n-1)$, $k = 1, 2, \ldots, N$, in (3). This assumption makes the analysis more tractable and does not affect the behavior of the proposed diffusion algorithm, as observed by simulations. Besides, it is only applied for the computation of the combined estimates $\mathbf{w}_k(n)$ [Eq. (3)] and it is not employed in the updating of the local estimates $\boldsymbol{\psi}_k(n)$ [Eq. (2)].

For the analysis of adaptive filters, it is customary to define weight-error vectors. In this case, these definitions should be done both for the local and combined estimates of each node: $\widetilde{\boldsymbol{\psi}}_k(n) \triangleq \mathbf{w}_{\mathrm{o}}(n) - \boldsymbol{\psi}_k(n)$ and $\widetilde{\mathbf{w}}_k(n) \triangleq \mathbf{w}_{\mathrm{o}}(n) - \mathbf{w}_k(n)$, for $k = 1, 2, \ldots, N$.

A-priori errors can be defined as filter errors in excess of the minimum value given by the noise:

$$\varepsilon_k(n) = e_k(n) - v_k(n) = \mathbf{u}_k^T(n)\widetilde{\boldsymbol{\psi}}_k(n-1), \tag{21}$$

using local estimates, and

$$\check{\varepsilon}_{k,\ell}(n) = \check{e}_{k,\ell}(n) - v_k(n) = \mathbf{u}_k^T(n)\widetilde{\mathbf{w}}_\ell(n-1), \tag{22}$$

for combined estimates, for $k = 1, 2, \ldots, N$, and $\ell \in \mathcal{N}_k$, where $\check{e}_{k,\ell}(n) = d_k(n) - \check{y}_{k,\ell}(n)$.

As a figure of merit of the performance of each node, we will use the local mean-square deviation (MSD), which is normally defined as $\mathrm{MSD}_k(n) = \mathrm{E}\{\|\widetilde{\mathbf{w}}_k(n)\|^2\}$. To assess the performance of the whole network, we will use the network MSD, defined as the average of the MSDs of all nodes, i.e.,

$$\mathrm{MSD}(n) = \frac{1}{N}\sum_{k=1}^{N}\mathrm{MSD}_k(n). \tag{23}$$

3.2. Mean-square analysis

Subtracting both sides of (3) from $\mathbf{w}_{\mathrm{o}}(n)$, using (20), and applying Assumption **A4**, we can approximate the weight-error vectors of the combined estimates as

$$\begin{aligned}
\widetilde{\mathbf{w}}_k(n) - \mathbf{q}(n) &= \overbrace{\left[c_{kk}(n) + \sum_{\ell\in\bar{\mathcal{N}}_k} c_{\ell k}(n)\right]}^{=1}\mathbf{w}_{\mathrm{o}}(n-1) - c_{kk}(n)\boldsymbol{\psi}_k(n) \\
&\quad - \sum_{\ell\in\bar{\mathcal{N}}_k} c_{\ell k}(n)\mathbf{w}_\ell(n-1) \\
&= c_{kk}(n)[\mathbf{w}_{\mathrm{o}}(n-1) - \boldsymbol{\psi}_k(n)] + \sum_{\ell\in\bar{\mathcal{N}}_k} c_{\ell k}(n)\widetilde{\mathbf{w}}_\ell(n-1) \\
&\approx c_{kk}(n)\widetilde{\boldsymbol{\psi}}_k(n-1) + \sum_{\ell\in\bar{\mathcal{N}}_k} c_{\ell k}(n)\widetilde{\mathbf{w}}_\ell(n-1).
\end{aligned} \tag{24}$$

Premultiplying both sides of (24) by their transposes, we obtain

$$\begin{aligned}\|\widetilde{\mathbf{w}}_k(n)\|^2 + \|\mathbf{q}(n)\|^2 - 2\mathbf{q}^T(n)\widetilde{\mathbf{w}}_k(n) \approx\ & c_{kk}^2(n)\|\widetilde{\boldsymbol{\psi}}_k(n-1)\|^2 \\ &+ \sum_{\ell\in\bar{\mathcal{N}}_k}\sum_{m\in\bar{\mathcal{N}}_k} c_{\ell k}(n)c_{mk}(n)\widetilde{\mathbf{w}}_\ell^T(n-1)\widetilde{\mathbf{w}}_m(n-1) \\ &+ 2\sum_{\ell\in\bar{\mathcal{N}}_k} c_{kk}(n)c_{\ell k}(n)\widetilde{\boldsymbol{\psi}}_k^T(n-1)\widetilde{\mathbf{w}}_\ell(n-1). \end{aligned} \tag{25}$$

Taking expectations on both sides of (25) and using **A1**, we arrive at

$$\begin{aligned}\mathrm{MSD}_k(n) \triangleq \mathrm{E}\{\|\widetilde{\mathbf{w}}_k(n)\|^2\} \approx\ & \mathrm{E}\left\{c_{kk}^2(n)\|\widetilde{\boldsymbol{\psi}}_k(n-1)\|^2\right\} \\ &+ \sum_{\ell\in\bar{\mathcal{N}}_k}\sum_{m\in\bar{\mathcal{N}}_k} \mathrm{E}\left\{c_{\ell k}(n)c_{mk}(n)\widetilde{\mathbf{w}}_\ell^T(n-1)\widetilde{\mathbf{w}}_m(n-1)\right\} \\ &+ 2\sum_{\ell\in\bar{\mathcal{N}}_k} \mathrm{E}\left\{c_{kk}(n)c_{\ell k}(n)\widetilde{\boldsymbol{\psi}}_k^T(n-1)\widetilde{\mathbf{w}}_\ell(n-1)\right\} - \mathrm{E}\{\|\mathbf{q}(n)\|^2\}. \end{aligned} \tag{26}$$

Defining the cross-covariance matrices of the local, combined, and local-combined weight-error vectors, i.e.,

$$\mathbf{S}_{\ell m}(n) \triangleq \mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell(n)\widetilde{\boldsymbol{\psi}}_m^T(n)\}, \tag{27}$$

$$\mathbf{W}_{\ell m}(n) \triangleq \mathrm{E}\{\widetilde{\mathbf{w}}_\ell(n)\widetilde{\mathbf{w}}_m^T(n)\}, \tag{28}$$

$$\mathbf{X}_{\ell m}(n) \triangleq \mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell(n)\widetilde{\mathbf{w}}_m^T(n)\}, \tag{29}$$

we can compute

$$\mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell^T(n)\widetilde{\boldsymbol{\psi}}_m(n)\} = \mathrm{Tr}[\mathbf{S}_{\ell m}(n)], \tag{30}$$

$$\mathrm{E}\{\widetilde{\mathbf{w}}_\ell^T(n)\widetilde{\mathbf{w}}_m(n)\} = \mathrm{Tr}[\mathbf{W}_{\ell m}(n)], \tag{31}$$

$$\mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell^T(n)\widetilde{\mathbf{w}}_m(n)\} = \mathrm{Tr}[\mathbf{X}_{\ell m}(n)], \tag{32}$$

where $\mathrm{Tr}(\cdot)$ stands for the trace of a matrix.

Thus, under Assumption **A3** and using (30)-(32), (26) can be rewritten as

$$\begin{aligned}\mathrm{MSD}_k(n) \approx\ & \mathrm{E}\left\{c_{kk}^2(n)\right\}\mathrm{Tr}[\mathbf{S}_{kk}(n-1)] + \sum_{\ell\in\bar{\mathcal{N}}_k}\sum_{m\in\bar{\mathcal{N}}_k} \mathrm{E}\left\{c_{\ell k}(n)c_{mk}(n)\right\}\mathrm{Tr}[\mathbf{W}_{\ell m}(n-1)] \\ &+ 2\sum_{\ell\in\bar{\mathcal{N}}_k} \mathrm{E}\left\{c_{kk}(n)c_{\ell k}(n)\right\}\mathrm{Tr}[\mathbf{X}_{k\ell}(n-1)] - \mathrm{Tr}(\mathbf{Q}). \end{aligned} \tag{33}$$

The network MSD can be estimated theoretically from the expression above by averaging the local MSD of all network nodes.

To complete the analysis, we must obtain analytical expressions for $\mathbf{S}_{\ell m}(n)$, $\mathbf{W}_{\ell m}(n)$, $\mathbf{X}_{\ell m}(n)$, and $\mathrm{E}\{c_{\ell k}(n)c_{mk}(n)\}$. Recursions for the cross-covariance matrices are provided below, with the proofs given in Appendix A.

$$\begin{aligned}\mathbf{S}_{\ell m}(n) \approx{}& \mathbf{S}_{\ell m}(n-1) - \overline{\mu}_m \mathbf{S}_{\ell m}(n-1)\,\mathbf{R}_{mm} - \overline{\mu}_\ell \mathbf{R}_{\ell\ell}\,\mathbf{S}_{\ell m}(n-1) \\ &+ \overline{\mu}_\ell \overline{\mu}_m \mathbf{R}_{\ell\ell}\mathbf{S}_{\ell m}(n-1)\mathbf{R}_{mm} + \mathbf{Q},\ (\ell \neq m) \qquad (34)\\ \mathbf{S}_{\ell\ell}(n) \approx{}& \mathbf{S}_{\ell\ell}(n-1) - \overline{\mu}_\ell \Big[\mathbf{S}_{\ell\ell}(n-1)\,\mathbf{R}_{\ell\ell} + \mathbf{R}_{\ell\ell}\,\mathbf{S}_{\ell\ell}(n-1)\Big] \\ &+ \overline{\mu}_\ell^2 \frac{M-2}{M-4}\Big[2\mathbf{R}_{\ell\ell}\mathbf{S}_{\ell\ell}(n-1)\mathbf{R}_{\ell\ell} + \mathbf{R}_{\ell\ell}\mathrm{Tr}(\mathbf{S}_{\ell\ell}(n-1)\mathbf{R}_{\ell\ell})\Big] \\ &+ \overline{\mu}_\ell^2 \frac{M-2}{M-4}\sigma_\ell^2 \mathbf{R}_{\ell\ell} + \mathbf{Q}, \qquad (35)\\ \mathbf{X}_{\ell m}(n) \approx{}& \mathrm{E}\{c_{mm}(n)\}[\mathbf{I} - \bar{\mu}_\ell \mathbf{R}_{\ell\ell}]\mathbf{S}_{\ell m}(n-1) \\ &+ [\mathbf{I} - \bar{\mu}_\ell \mathbf{R}_{\ell\ell}] \sum_{p\in\bar{\mathcal{N}}_m} \mathrm{E}\{c_{pm}(n)\}\mathbf{X}_{\ell p}(n-1) + \mathbf{Q}, \qquad (36)\\ \mathbf{W}_{\ell m}(n) \approx{}& \mathrm{E}\{c_{\ell\ell}(n)c_{mm}(n)\}\mathbf{S}_{\ell m}(n-1) \\ &+ \sum_{p\in\bar{\mathcal{N}}_\ell}\sum_{r\in\bar{\mathcal{N}}_m} \mathrm{E}\{c_{p\ell}(n)c_{rm}(n)\}\mathbf{W}_{pr}(n-1) \\ &+ \sum_{r\in\bar{\mathcal{N}}_m} \mathrm{E}\{c_{\ell\ell}(n)c_{rm}(n)\}\mathbf{X}_{\ell r}(n-1) \\ &+ \sum_{p\in\bar{\mathcal{N}}_\ell} \mathrm{E}\{c_{p\ell}(n)c_{mm}(n)\}\mathbf{X}_{mp}^T(n-1) + \mathbf{Q}, \qquad (37)\end{aligned}$$

where we have defined

$$\overline{\mu}_k \triangleq \frac{\mu_k}{\sigma_{u_k}^2 (M-2)}, \qquad (38)$$

with $\sigma_{u_k}^2$ being the variance of the input signal at node k, with $k = 1, 2, \ldots, N$.

3.3. Network MSD for the optimal combination weights

Obtaining a theoretical model of the cross-correlation between the different combination weights is a very difficult task given their dependencies with the combined estimation vectors, which are shared among network nodes at each iteration. For this reason, we limit ourselves to modeling the optimal values of these combination parameters. Thus, when using in (33) the approximation

$$\mathrm{E}\{c_{\ell k}(n)c_{mk}(n)\} \approx c_{\ell k,\mathrm{o}}(n)c_{mk,\mathrm{o}}(n), \quad l,k,m \in \mathcal{N}_k \qquad (39)$$

where $c_{\ell k,\mathrm{o}}(n)$ stands for the optimal value of $c_{\ell k}(n)$, we will be obtaining a theoretical value of the minimum MSD that can be achieved at each node. In practice, the adjustment of the combination weights would be subject to some non-negligible variance

due to the employment of finite-length windows, and thus the real MSD will exceed this optimal value. Nevertheless, by obtaining this theoretical limit we can get some insight about the quality of the LS-based update scheme for the combination weights, and the level of residual error it introduces.

Starting from (7), that defines the optimal combination weights $\bar{\mathbf{c}}_{k,\mathrm{o}}(n)$, and following derivations similar to (10)-(18), it is straightforward to show that

$$\bar{\mathbf{c}}_{k,\mathrm{o}}(n) = \mathbf{P}_{k,\mathrm{o}}^{-1}(n)\mathbf{z}_{k,\mathrm{o}}(n). \tag{40}$$

Now, we rewrite the components of vector $\tilde{\mathbf{y}}_k(n)$ in a more convenient way using (11) and (22), i.e.

$$\tilde{y}_{k,p}(n) = \check{y}_{k,p}(n) - y_k(n) = e_k(n) - \check{e}_{k,p}(n) = \varepsilon_k(n) - \check{\varepsilon}_{k,p}(n). \tag{41}$$

Particularizing the expression above for $p = \ell$ and $p = m$, with $\ell, m \in \bar{\mathcal{N}}_k$, multiplying the results, and taking expectations, we have the values of covariance matrix $\mathrm{E}\{\tilde{\mathbf{y}}_k(n)\tilde{\mathbf{y}}_k^T(n)\}$ for different neighbors ℓ and m of node k:

$$\begin{aligned}\mathrm{E}\{\tilde{y}_{k,\ell}(n)\tilde{y}_{k,m}(n)\} = \mathrm{E}\{\varepsilon_k^2(n)\} + \mathrm{E}\{\check{\varepsilon}_{k,\ell}(n)\check{\varepsilon}_{k,m}(n)\} \\ - \mathrm{E}\{\varepsilon_k(n)\check{\varepsilon}_{k,\ell}(n)\} - \mathrm{E}\{\varepsilon_k(n)\check{\varepsilon}_{k,m}(n)\}.\end{aligned} \tag{42}$$

To proceed further, we estimate the terms in (42) recalling that Assumption **A2** implies that $\widetilde{\boldsymbol{\psi}}_k(n-1)$ and $\widetilde{\mathbf{w}}_\ell(n-1)$ are independent of $\mathbf{u}_k(n)$ for all $\ell \in \bar{\mathcal{N}}_k$, which leads to

$$\mathrm{E}\{\varepsilon_k^2(n)\} = \mathrm{E}\{\mathbf{u}_k^T(n)\widetilde{\boldsymbol{\psi}}_k(n-1)\widetilde{\boldsymbol{\psi}}_k^T(n-1)\mathbf{u}_k(n)\} \approx \mathrm{Tr}[\mathbf{S}_{kk}(n-1)\mathbf{R}_{kk}], \tag{43}$$

$$\mathrm{E}\{\check{\varepsilon}_{k,\ell}(n)\check{\varepsilon}_{k,m}(n)\} = \mathrm{E}\{\mathbf{u}_k^T(n)\widetilde{\mathbf{w}}_\ell(n-1)\widetilde{\mathbf{w}}_m^T(n-1)\mathbf{u}_k(n)\} \approx \mathrm{Tr}[\mathbf{W}_{\ell m}(n-1)\mathbf{R}_{kk}] \tag{44}$$

$$\mathrm{E}\{\varepsilon_k(n)\check{\varepsilon}_{k,\ell}(n)\} = \mathrm{E}\{\mathbf{u}_k^T(n)\widetilde{\mathbf{w}}_\ell(n-1)\widetilde{\boldsymbol{\psi}}_k^T(n-1)\mathbf{u}_k(n)\} \approx \mathrm{Tr}[\mathbf{X}_{k\ell}(n-1)\mathbf{R}_{kk}] \tag{45}$$

with $\ell, m \in \bar{\mathcal{N}}_k$.

Inserting these expressions back in (42), the (f,g)-th entry of matrix $\mathbf{P}_{k,\mathrm{o}}(n)$ is given by

$$[\mathbf{P}_{k,\mathrm{o}}(n)]_{f,g} = \mathrm{Tr}\left\{[\mathbf{S}_{kk}(n-1) + \mathbf{W}_{\bar{b}_k^{(f)}\bar{b}_k^{(g)}}(n-1) - \mathbf{X}_{k\bar{b}_k^{(f)}}(n-1) - \mathbf{X}_{k\bar{b}_k^{(g)}}(n-1)]\mathbf{R}_{kk}\right\} \tag{46}$$

where $f, g = 1, 2, \cdots, \bar{N}_k$, and column vector $\bar{\mathbf{b}}_k$ is again used for notational convenience.

Similarly, we can compute the components of $\mathbf{z}_{k,\mathrm{o}}(n)$ from

$$\mathrm{E}\{e_k(n)\tilde{y}_{k,\ell}(n)\} = \mathrm{E}\{[\varepsilon_k(n) + v_k(n)][\varepsilon_k(n) - \check{\varepsilon}_{k,p}(n)]\}. \tag{47}$$

Since the noise term is independent from the *a priori* errors, and exploiting again (43) and (45), we arrive at

$$[\mathbf{z}_{k,\mathrm{o}}(n)]_f = \mathrm{Tr}\left\{[\mathbf{S}_{kk}(n-1) - \mathbf{X}_{k\bar{b}_k^{(f)}}(n-1)]\mathbf{R}_{kk}\right\}. \tag{48}$$

To validate the accuracy of the analysis, we have carried out a preliminary experiment with the network of three interconnected nodes represented in Fig. 2. The step sizes of the different nodes are $\mu_1 = \mu_3 = 0.1$ and $\mu_2 = 1$. Using different step sizes provides a way to introduce diversity among network nodes, and can be used as a way to improve the network capabilities in terms of convergence and tracking performance. More details about the simulation setup for this simple network will be given in Subsection 5.1.

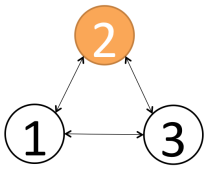


Figure 2: Network topology for the experiments of Subsections 3.3 and 5.1. Shaded node 2 is adapted with $\mu_2 = 1$ and the rest with $\mu_1 = \mu_3 = 0.1$.

Fig. 3 displays the curves for the network performance, both the simulated MSD estimated as an average over 500 runs, and the theoretical MSD that is computed from the theoretical model. As expected, the MSD from the real network exceeds to some extent the values predicted by the theoretical model, mostly because the estimation of combination weights using finite-length windows is subject to non-zero (though small) variance, a circumstance which is not taken into account by the theoretical model. Nevertheless, the deviation is not very significant and, more importantly, the model predicts well the qualitative behavior of the network MSD and the time instants where the MSD has roughly converged to -20 dB and for the transition from -20 to -40 dB.

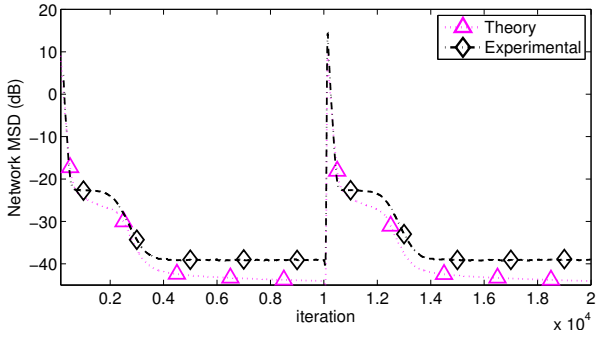


Figure 3: Simulated and theoretical MSD for a network with three nodes with different step sizes.

We should notice that the conditioning of matrix $\mathbf{P}_{k,\mathrm{o}}(n)$ strongly depends on the initializations of the cross-covariance matrices $\mathbf{W}_{\bar{b}_k^{(f)}\bar{b}_k^{(g)}}$, $\mathbf{X}_{k\bar{b}_k^{(f)}}$, and $\mathbf{X}_{k\bar{b}_k^{(g)}}$. Initializing the local weights with the null vector, $\boldsymbol{\psi}_k(-1) = \mathbf{0}$, is equivalent to initialize the cross-covariance matrices $\mathbf{S}_{kk}$, $\mathbf{W}_{\bar{b}_k^{(f)}\bar{b}_k^{(g)}}$, $\mathbf{X}_{k\bar{b}_k^{(f)}}$, and $\mathbf{X}_{k\bar{b}_k^{(g)}}$ with $\mathbf{w}_{\mathrm{o}}(-1)\mathbf{w}_{\mathrm{o}}^T(-1)$ [see Eqs. (27)-(29)]. Consequently, according to Eq. (46), $\mathbf{P}_{k,\mathrm{o}}(n)$ would become the null matrix, causing the divergence of the model due to the computation of the inverse $\mathbf{P}_{k,\mathrm{o}}^{-1}(n)$ in (40). To avoid an ill-conditioning of this matrix and divergence in the calculus of the model, it is important to initialize these cross-covariance matrices with suitable values and regularize the matrix $\mathbf{P}_{k,\mathrm{o}}(n)$. To obtain the results of Fig. 3, the model was initialized as follows $\mathbf{S}_{kk}(-1) = \mathbf{w}_{\mathrm{o}}(-1)\mathbf{w}_{\mathrm{o}}^T(-1)$, $\mathbf{W}_{\bar{b}_k^{(f)}\bar{b}_k^{(g)}}(-1) = w_g\mathbf{w}_{\mathrm{o}}(-1)\mathbf{w}_{\mathrm{o}}^T(-1)$, $\mathbf{X}_{k\bar{b}_k^{(f)}}(-1) = x_f\mathbf{w}_{\mathrm{o}}(-1)\mathbf{w}_{\mathrm{o}}^T(-1)$, and $\mathbf{X}_{k\bar{b}_k^{(g)}}(-1) = x_g\mathbf{w}_{\mathrm{o}}(-1)\mathbf{w}_{\mathrm{o}}^T(-1)$, where w_g, x_f, and x_g are positive constants randomly chosen in the interval $[0.01,\ 0.1]$. Additionally, $\mathbf{P}_{k,\mathrm{o}}(n)$ was regularized by loading its main diagonal with a small constant (e.g., 10^{-7}).

4. Hybrid Scheme

Preliminary experiments carried out in [1] showed that the stationary steady-state network MSD of the proposed scheme is slightly worse than that achieved by other diffusion schemes with adaptive combiners. To illustrate this, we recur again to the 3-node network of Fig. 2. In Fig. 4, we represent the MSD of the node with large step size (Node 2) for the proposed ADN-LS algorithm and compare it to the algorithm from [10] that also implements adaptive combination weights in diffusion networks (denominated as Adaptive Combination Weights (ACW) in [12]) and that can be considered

the state of the art. More details about the simulation setup can be found in Subsection 5.1.

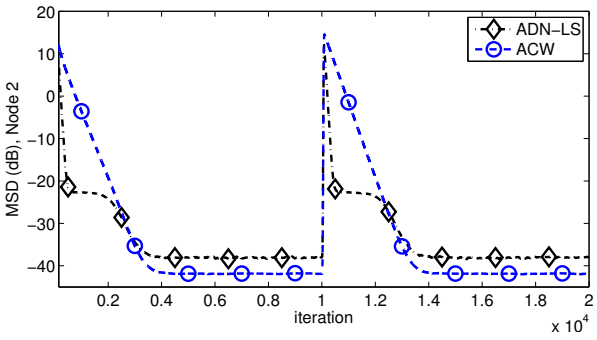


Figure 4: MSD evolution of the ADN-LS and the ACW schemes at Node 2 for the simple network setup described in Subsection 5.1.

As it can be seen, the ACW scheme actually degrades the convergence of this fast node, while ADN-LS keeps a very fast convergence. This non-desired behavior of ACW is partly caused by the feedback of weights into the nodes update equation. Being designed under approximations that mostly hold in steady state, during convergence ACW tends to give large weights to slowly-varying nodes, and this implies also that the weights coming from slow nodes are incorporated in the adaptation phase of the fast node, slowing down its convergence. Indeed, this was the main reason that motivated us to keep pure local estimates at each node, since precisely one of the goals of ADN-LS is to exploit the complementary properties of nodes adapted with different step sizes.

On the downside, although both schemes outperform the case with no cooperation, it is clear that ACW is able to obtain a more reduced steady-state error. A similar behavior is observed for the other nodes in the network. It is worth mentioning that the theoretical model derived in the previous section predicted an MSD level slightly below -40 dB, that is similar to the MSD achieved by the ACW algorithm. The main reason for this loss of performance of our proposal is the noise in the estimation of $c_{\ell k,\mathrm{o}}(n)$. For this reason, we present an approach that can benefit from the complementary properties of ADN-LS and ACW, and which is a straightforward application of schemes available in the literature for the combination of adaptive systems.

More specifically, in this section we present a strategy based on a hybrid scheme that combines the estimates obtained by both diffusion strategies, ADN-LS and ACW.

Obviously, the cost we pay is an increased computational complexity and communication cost. If we denote by $\mathbf{w}_k^{\mathrm{LS}}(n)$ and $\mathbf{w}_k^{\mathrm{ACW}}(n)$ the combined weights of the ADN-LS and ACW strategies at node k, the hybrid approach would obtain the overall weights as

$$\mathbf{w}_k^{\mathrm{HB}}(n) = \alpha_k(n)\mathbf{w}_k^{\mathrm{LS}}(n) + [1-\alpha_k(n)]\mathbf{w}_k^{\mathrm{ACW}}(n), \tag{49}$$

where $\alpha_k(n)$ is a combination parameter to be updated in order to optimize the overall performance.

Eq. (49) can be seen as an adaptive combination of two adaptive filters. Different strategies have been proposed in the literature to adaptively mix the weights of two adaptive filters, including both convex and affine combinations [20, 24, 35, 36]. Here, we restrict ourselves to the convex combination case, $\alpha_k(n) \in [0,1]$, since adaptation rules for the convex case are normally easier to adjust with respect to parameter selection [25].

In order to adapt $\alpha_k(n)$ at each node, we use a normalized scheme which is a slight variation of a previously proposed rule [21, 37]. Instead of directly adapting $\alpha_k(n)$, an auxiliary parameter $a_k(n)$ is updated seeking to minimize the power of the combined error $e_k^{\mathrm{HB}}(n) = d_k(n) - \mathbf{u}_k^T(n)\mathbf{w}_k^{\mathrm{HB}}(n)$. Parameters $\alpha_k(n)$ and $a_k(n)$ are univocally related by means of the sigmoid function,

$$\alpha_k(n) = \mathrm{sgm}\{a_k(n)\} = \frac{1}{1+e^{-a_k(n)}}. \tag{50}$$

We follow a normalized stochastic gradient algorithm to minimize the power of $e_k^{\mathrm{HB}}(n)$, giving rise to:

$$\begin{aligned} a_k(n+1) &= a_k(n) - \frac{\mu_{a,k}}{p_k(n)} \frac{\partial [e_k^{\mathrm{HB}}(n)]^2}{\partial a_k(n)} \\ &= a_k(n) + \frac{\mu_{a,k}}{p_k(n)} e_k^{\mathrm{HB}}(n)\mathbf{u}_k^T(n)[\mathbf{w}_k^{\mathrm{LS}}(n) - \mathbf{w}_k^{\mathrm{ACW}}(n)] \frac{\mathrm{d}\alpha_k(n)}{\mathrm{d}a_k(n)}, \end{aligned} \tag{51}$$

where $\mu_{a,k}$ is a step size that governs the update, and $p_k(n+1) = \beta p_k(n) + (1-\beta)[e_k^{\mathrm{HB}}(n)]^2$, with $0 \ll \beta < 1$, is a rough estimation of the power of the combined error $e_k^{\mathrm{HB}}(n)$. This normalization constitutes a slightly modification of the original adaptation rule proposed in [25] and enhances the update of $a_k(n)$ when both component schemes behave similarly.

The benefits of the sigmoid function (50) are twofold: 1) It restricts mixing parameters to range $[0,1]$, and 2) the derivative of the sigmoid function that appears in (51) reduces the amount of gradient noise when $\alpha_k(n)$ is close to the limits. Specific details about this adaptation scheme can be found in previous contributions [20, 21, 37].

The advantages of the hybrid scheme will be shown in the experiment section, both for stationary and tracking situations.

4.1. Computational and Communication Costs

Here, we analyze the algorithm complexity in terms of computational and communication costs. In Table 2, we display the number of floating-point operations needed for the three different steps of the algorithm: adaptation, combination and computation of combination coefficients. Both ACW [10] and ADN-LS, have equivalent complexity for the adaptation and combination steps; consequently we compare the cost of adapting the network combiners. Focusing on the number of multiplications required by the algorithms, ACW needs roughly $N_k M$ products per iteration, whereas ADN-LS requires around $(N_k - 1)M + 6N_k^2$ products[4]. Thus, both algorithms have similar complexity for a typical case with $N_k \ll M$, while the hybrid scheme requires roughly twice as much computational cost, where we have ignored the very reduced number of operations required by the combination.

Similarly, with respect to communication costs, both ACW and ADN-LS require exactly the same exchange of information among nodes (for a common topology), whereas the hybrid scheme duplicates that demand for the simultaneous diffusion of $\mathbf{w}^{\mathrm{ACW}}(n)$ and $\mathbf{w}^{\mathrm{LS}}(n)$. Nevertheless, some strategies could be applied to minimize these requirements. For example, for some nodes $\alpha_k(n) \approx 0$ or $\alpha_k(n) \approx 1$ most of the time; consequently in these cases we could avoid transmission of the irrelevant weight vector, without seriously affecting the network MSD.

Strategies for reducing the computational cost could also be designed, e.g., by following the approach in [23] that instead of adapting two filters in parallel, adapts just one of them and the difference filter. Since the latter usually has lower dynamic range, it can be implemented using a smaller number of bits. The convex combination of ADN-LS and ACW could also take advantage of this strategy to reduce computational and communication costs.

[4]We assume that the inversion of matrix $\mathbf{P}_k(n)$ at each node is implemented using matrix inversion lemma [15].

Table 2: Computational cost of ACW and the proposed ADN-LS scheme for every node k and iteration i.

	Products	Sums
Adapt	$2M+1$	$2M$
Combine	M	$N_k M$
Combiners ACW	$N_k M + 2N_k$	$2N_k M + N_k$
Combiners ADN-LS	$(N_k-1)M + 6N_k^2$ $-8N_k + 2$	$(N_k-1)M + 6N_k^2$ $-12N_k + 5$

5. Simulation results

The experiments we present in this section pursue two objectives. First, providing empirical evidence of the performance of the proposed scheme, ADN-LS, compared with the state-of-the-art ACW algorithm[5][10, 12]. Second, showing that the hybrid algorithm that blends both schemes using a convex combination is able to obtain the best features of each scheme. To do so, we consider both a very simple network (Fig. 2) to better analyze the behavior of the new rule for adapting the combination weights, and a complex network (Fig. 6) to study the performance of the methods in more challenging situations, both in stationary and tracking scenarios.

The following settings are common for all experiments. We employ a rectangular window of length $L = 100$ for ADN-LS in all cases, while for ACW the step size ν is empirically selected equal to $\nu = 0.01$ in order to obtain its best performance. Unless otherwise stated, provided results have been averaged over 500 independent realizations.

5.1. Simple network

We start by considering a very simple network topology composed of three interconnected nodes, as shown in Fig. 2. Such a simple configuration allows a more direct study of the update rules for the combiners, isolating them from the effects of diffusion in more complex networks.

In the first experiment, we focus on the stationary case where $\mathbf{w}_o$ is a vector of length $M = 50$ with random components taken from a uniform distribution between

[5]We do not compare our scheme to the algorithm of [18], since that algorithm is not able to reconverge, which is a fundamental property in adaptive systems.

-1 and 1. To study the ability of the algorithms to reconverge, an abrupt change is introduced at $n = 10000$. Input regressors and observation noises are independent across the network nodes. Furthermore, the regressors $\mathbf{u}_k(n)$, $k = 1, 2, 3$, follow a multidimensional Gaussian distribution with zero mean and covariance matrix equal to the identity ($\mathbf{I}$). The observation noise $v_k(n)$ is generated independently of $\mathbf{u}_k(n)$ and also follows a Gaussian distribution with zero mean and variances selected at random from interval $[10^{-3}, 10^{-2}]$ to get a different SNR at each node.

We consider NLMS adaptation for the local updates at each node, with step sizes $\mu_2 = 1$ (shaded node in Fig. 2) and $\mu_1 = \mu_3 = 0.1$. Using different step sizes at each node can be considered as a way of introducing diversity and can improve the convergence and tracking capabilities of adaptive networks, as we show later. In fact, this possibility has already been suggested previously in several works [3, 5, 6, 7, 9], although the experimental work in these papers was restricted to the case where all nodes use the same step size. The learning rate of the combination rule for the hybrid scheme in (51) is set to $\mu_{a,k} = 1$ for all nodes.

We show in Fig. 5-(a) the MSD evolution of the local estimates of all three nodes, i.e., $\mathrm{MSD}_k^{\mathrm{local}}(n) = \mathrm{E}\{\|\mathbf{w}_{\mathrm{o}} - \boldsymbol{\psi}_k(n)\|^2\}$. As expected, node 2 shows a much faster convergence but achieves also a larger steady-state misadjustment than the rest of nodes, as it uses a larger step size. In Fig. 5-(b), we show the MSD evolution for the combined estimates of this second node for the three considered schemes. We can see that when using the ADN-LS combination rule, the node follows the initial convergence associated to its own step size, and then takes advantage of the smaller steady-state MSD achieved by nodes 1 and 3. However, we observe that application of the ACW strategy makes this node behave similarly to nodes with slow adaptation, degrading the fast convergence that would be associated with its own step size. This is due to the fact that ACW attempts to minimize the variance of the combined estimate, thus favoring nodes with slow adaptation. Note that this situation is likely to occur when we have different step sizes in the network. Finally, the hybrid scheme is able to converge as fast as the ADN-LS algorithm, while achieving the lower steady-state MSD of the ACW algorithm. This behavior can also be explained from the evolution of the mixing parameters depicted in Fig. 5-(c). Finally, we should mention that for this very simple network the MSD of the combined estimations of all nodes are very similar, so that network MSD curves are also very similar to those depicted in Fig. 5-(b).

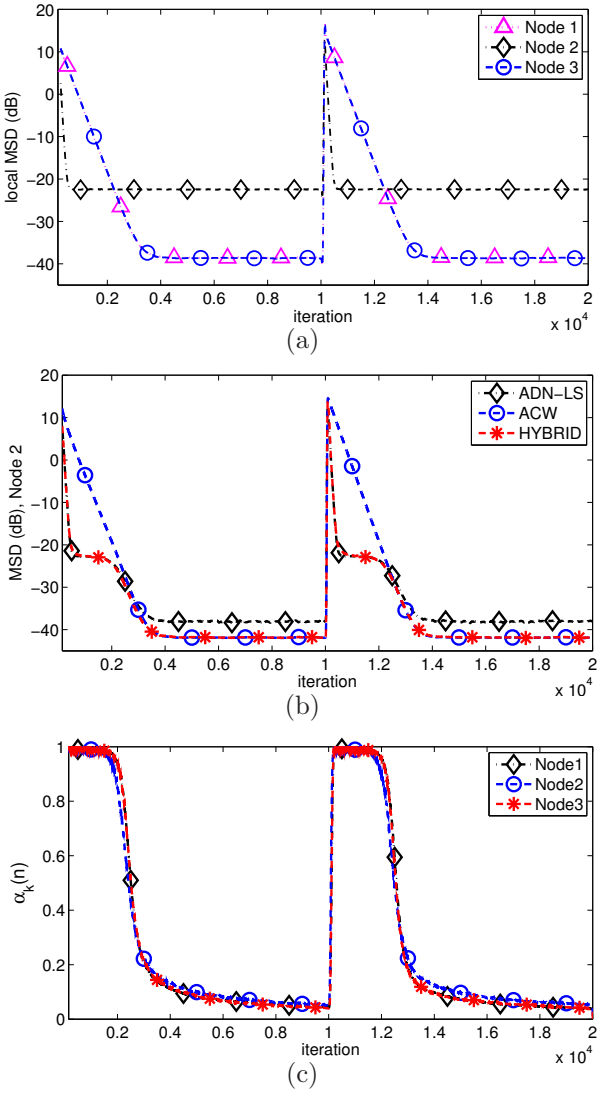


Figure 5: Algorithm behavior for the simple network case in a stationary scenario. (a) MSD of the local estimates $\boldsymbol{\psi}_k(n)$ at the different nodes; (b) MSD of combined estimates at Node 2 using ADN-LS, ACW, and the hybrid scheme; (c) Combination parameters $\alpha_k(n)$ for the 3 nodes.

5.2. Performing under a complex network

In this subsection, we analyze the performance of our algorithm in a more challenging scenario with a network composed of 15 nodes. The selected network is illustrated in Fig. 6-(a) and was previously used in [1] and [9]. Settings for the input regressors, observation noise, and unknown parameter vector are the same as in the previous case. The noise variances at each node are shown in Fig. 6-(b). All nodes apply NLMS rules for their adaptation phase; ten of the nodes were selected at random to use a step size

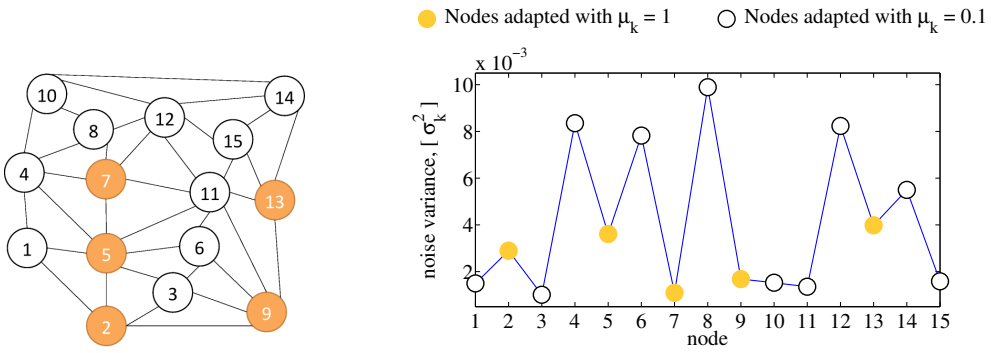


Figure 6: (a) Network topology for the experiments of Subsection 5.2. Shaded nodes are adapted with $\mu_k = 1$ and the rest with $\mu_k = 0.1$. (b) Noise power at each network node.

of $\mu_k = 0.1$, while the rest use $\mu_k = 1$ (shaded nodes in Fig. 6).

First of all, the influence of the length of the sliding window L in the performance of the ADN-LS algorithm is studied in Fig. 7. As explained at the end of Section 2, the length of the window is subject to a trade-off between convergence capabilities and steady-state performance (similarly to a standard recursive least squares filter). As we can see, a too long window, such as $L = 500$, results in poorer forgetting capabilities and therefore slower convergence. It should be noted that, although lower residual errors are achieved with longer windows, this characteristic is provided by the ACW component in the hybrid scheme. For this reason, in the following we choose a relatively short window ($L = 100$) to provide a very fast convergence to the proposed ADN-LS algorithm, and hence to the hybrid scheme.

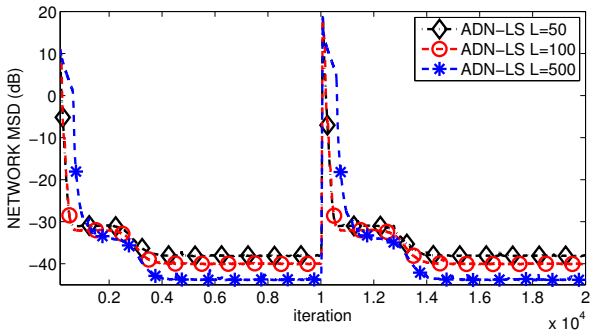
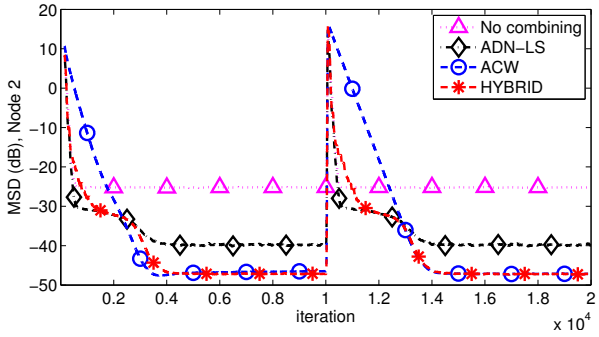


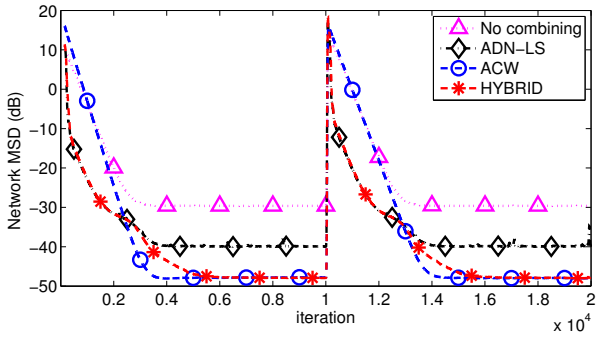
Figure 7: Network MSD of the ADN-LS diffusion network for three different window lengths.

Fig. 8 illustrates the convergence and steady-state behavior of the three methods. Subfig. 8-(a) displays the MSD for node 2 of the network, whereas Subfig. 8-(b) provides the average network MSD. The MSD for a case with no cooperation among nodes has also been included as a baseline for comparison. We can see that all diffusion schemes outperform this baseline, evidencing the advantages of information sharing among the nodes. As before, ADN-LS converges much faster than ACW, again due to the fact that ACW combiners favor the slow nodes in the network, while ADN-LS can exploit the convergence properties of fast nodes. In steady state, however, ACW outperforms the ADN-LS scheme, mainly because of the variance in the estimation of the combiners introduced by the LS adaptation with small sliding window.

When analyzing the performance of the hybrid scheme, some delay can be noticed in the switching between the ADN-LS and ACW schemes around $n = 4000$ and $n = 14000$, which is unavoidable as long as a stochastic gradient mechanism is used to adjust the combination parameters $\alpha_k(n)$. Nevertheless, such delay is normally not very significant, and we can state that the hybrid scheme is able to achieve simultaneously fast convergence (thanks to the ADN-LS scheme) and the smaller steady-state misadjustment of ACW. It is important to mention that, unlike other strategies for improving the performance of adaptive filters, the adjustment of the hybrid scheme does not require any *a priori* knowledge of the statistics of the scenario, such as nodes SNRs or $\mathrm{Tr}(\mathbf{Q})$.



(a)



(b)

Figure 8: Learning curves for the complex network and all studied schemes: (a) MSD for the combined estimates of node 2. (b) Network MSD.

We close this section by illustrating the tracking capabilities of our scheme. We assume in this case that the common vector of parameters of interest $\mathbf{w}_{\mathrm{o}}(n)$ varies according to the following tracking model taken from [15]:

$$\begin{cases} \mathbf{w}_{\mathrm{o}}(n) = \mathbf{w}_{\mathrm{o}} + \boldsymbol{\theta}(n) \\ \boldsymbol{\theta}(n) = 0.99\,\boldsymbol{\theta}(n-1) + \mathbf{q}(n), \end{cases} \tag{52}$$

where $\mathbf{q}(n)$ is a sequence of i.i.d. perturbations with zero mean and covariance matrix $\mathbf{Q}$, independent of the input regressors and output noise at every iteration. This model is slightly different from that of Equation (20) to avoid that covariance matrix of $\mathbf{w}_{\mathrm{o}}$ grows to infinity. In this scenario, $\mathrm{Tr}(\mathbf{Q})$ can be considered as a measurement of the speed of change of the unknown parameters. Therefore, we will select $\mathbf{Q} = \sigma_q^2\mathbf{I}$, changing the value of σ_q to control the speed of variation of the unknown parameter vector.

Fig. 9 shows the steady-state network MSD of the three considered schemes for

different speeds of changes in $\mathbf{w}_o(n)$, by averaging over 200 realizations and 800 iterations after convergence is completed. To get an easier comparison among algorithms, all results have been normalized with respect to the steady-state MSD of the ACW network. Thus, a positive value in the plot represents a gain over ACW, whereas negative values indicate degraded performance. When the changes in $\mathbf{w}_o(n)$ are slow, it is the ACW algorithm which provides a better behavior, whereas the ADN-LS algorithm shows better tracking capabilities for $\mathrm{Tr}(\mathbf{Q}) > 10^{-5}$. The hybrid scheme obtains the best performance overall, even (very slightly) outperforming both ADN-LS and ACW for a range of $\mathrm{Tr}(\mathbf{Q})$ around 10^{-5}.

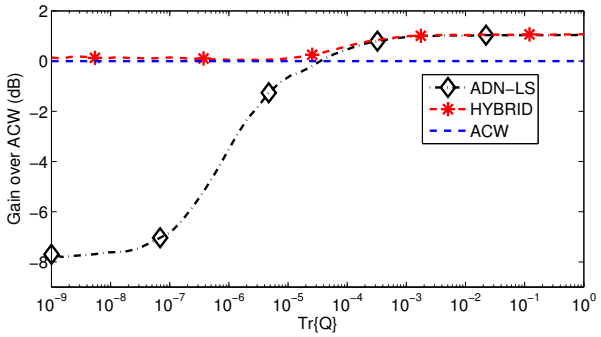


Figure 9: Tracking performance in terms of steady-state network MSD for the complex network and different $\mathrm{Tr}(\mathbf{Q})$. The MSDs of all schemes have been normalized with respect to the network MSD of the ACW algorithm.

To further illustrate tracking performance we show in Fig. 10 the convergence of network MSD for two different speeds of changes in $\mathbf{w}_o(n)$: (a) $\mathrm{Tr}(\mathbf{Q}) = 2\times10^{-5}$ (slow), and (b) $\mathrm{Tr}(\mathbf{Q}) = 0.1$ (fast). For slow changes, the steady-state MSD of all algorithms is approximately the same, although ADN-LS and the hybrid scheme provide significantly faster convergence. For fast changes [Fig. 10-(b)], the diffusion effect of the ACW algorithm is actually harmful, and makes all nodes behave similarly to NLMS with $\mu_k = 0.1$, degrading ACW performance even with respect to the baseline. In contrast, ADN-LS and the hybrid scheme show better convergence and similar steady-state tracking error than the baseline.

For this specific example, we should remark that according to Table 2, the number of products required at each iteration to update the combination weights of a node with six neighbors (the average neighborhood for this network) is 312 for ACW and

420 for ADN-LS. Consequently, as the matrices to invert are small, both ADN-LS and ACW have similar computational complexity.

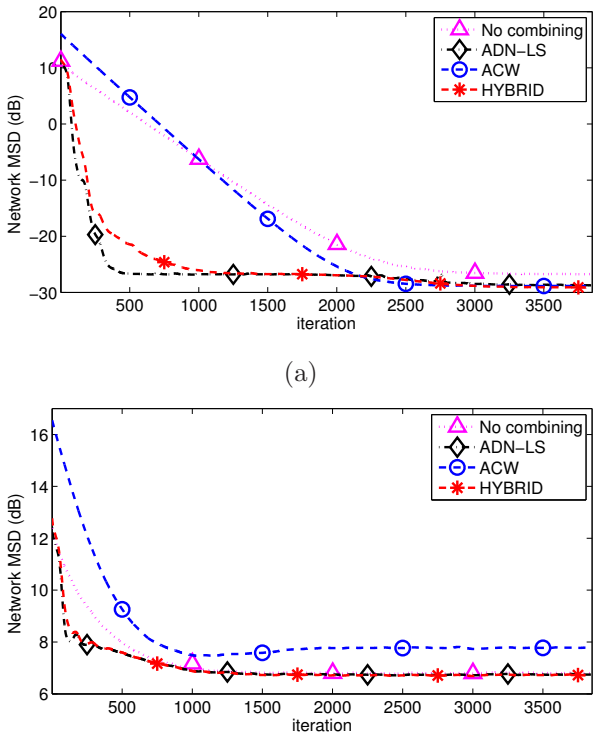


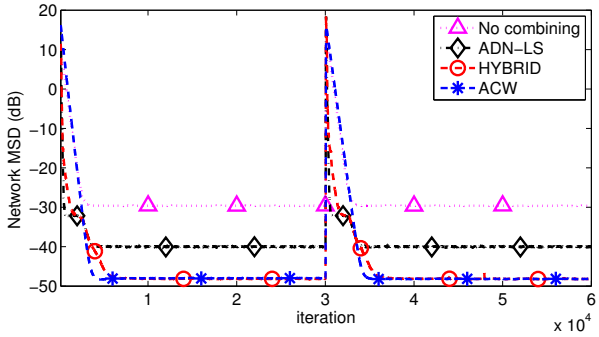
Figure 10: Network MSD evolution for a tracking situation with (a) $\mathrm{Tr}(\mathbf{Q}) = 2 \times 10^{-5}$, and (b) $\mathrm{Tr}(\mathbf{Q}) = 0.1$.

5.3. Performing under node failures

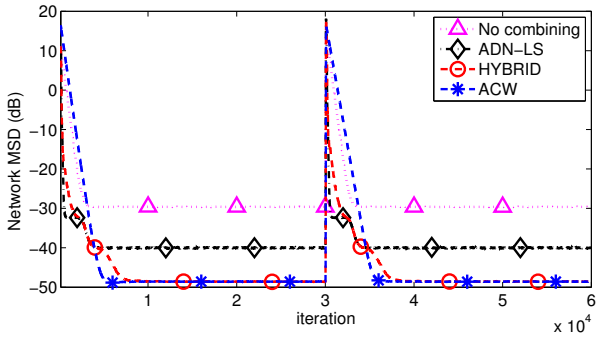
In the previous subsections, we have assumed that the nodes receive the estimates of all neighbors at every iteration. Here, we present a set of experiments allowing for node failure with a fixed probability. As a result of the failure of a node, its estimation is not transmitted to neighbors, and nodes carry out the combination step at every iteration by using the last received estimate from each neighbor.

In Figure 11, we depict the network MSD for different failure rates: (a) 10%, (b) 30%, and (c) 50%, considering a stationary scenario with similar settings to those employed in Subsection 5.2. In the light of these results, we can conclude that both ACW and ADN-LS remain quite robust to node failures in terms of steady-state performance. However, while ADN-LS convergence remains unaffected, ACW convergence deterio-

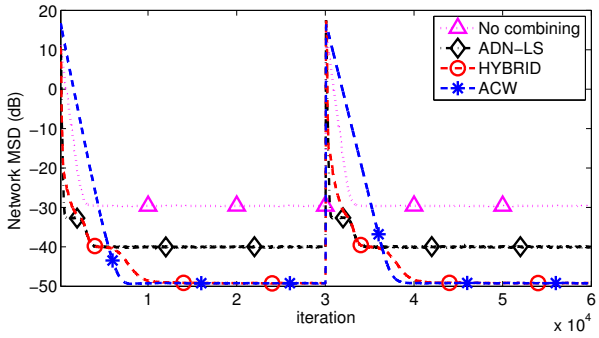
rates (being even slower than the noncooperative strategy for error rates of 30% and 50%). Finally, even under the node failures, the hybrid scheme is still able to exploit the faster convergence of ADN-LS and the smaller steady-state error of ACW to obtain an improved performance.



(a)



(b)



(c)

Figure 11: Network MSD for the complex network when nodes are subject to random failures at every iteration, with failure rate: (a) 10%, (b) 30%, and (c) 50%.

6. Conclusions

An important decision in the design of an adaptive network is how information is combined among neighboring nodes. In this paper, we have proposed a scheme in which each node combines a pure local estimation with the combined estimations received from its neighbors. To do that, each node adaptively calculates the combination weights minimizing a least-squares function with sliding window. A theoretical analysis of the proposed ADN-LS scheme has been derived, providing a valuable model both for the convergence and steady-state performance of the algorithm with optimal combiners.

Our algorithm outperforms a state-of-the-art scheme in terms of convergence and tracking, but its practical implementation suffers a slight degradation of its steady-state performance with respect to other adaptive solutions, mostly due to the variance in the estimation of the optimal combiners in the steady state. For this reason, we have also proposed a hybrid scheme based on the adaptive combination of ADN-LS with other state-of-the-art algorithm that employs adaptive combiners (ACW). The experiments also show the ability of the hybrid scheme to take advantage of the benefits of each of its components.

Future work will focus on developing more robust implementations of the ADN-LS scheme to reduce the noise in the estimations of the combiners, as well as on new diffusion strategies that take advantage of the combined estimations also during the local adaptation stage. In addition, to guarantee the feasible implementation and deployment of these schemes in real sensor network applications, the behavior of diffusion networks under nodes failures will be studied both theoretically and empirically. Finally, a very important future activity is the deployment of a physical sensor network that allows a realistic evaluation of the properties of our proposal. We will focus on the application of localization and tracking of targets in acoustic environments, where thanks to the good tracking and convergence properties of our scheme, a suitable performance is expected.

A. Recurrent expressions for the cross-variance matrices

In this appendix, we obtain recurrent expressions for $\mathbf{S}_{\ell m}(n) \triangleq \mathrm{E}\{\widetilde{\boldsymbol{\psi}}_{\ell}(n)\widetilde{\boldsymbol{\psi}}_{m}^{T}(n)\}$, $\mathbf{X}_{\ell m}(n) \triangleq \mathrm{E}\{\widetilde{\boldsymbol{\psi}}_{\ell}(n)\widetilde{\mathbf{w}}_{m}^{T}(n)\}$, and $\mathbf{W}_{\ell m}(n) \triangleq \mathrm{E}\{\widetilde{\mathbf{w}}_{\ell}(n)\widetilde{\mathbf{w}}_{m}^{T}(n)\}$, assuming $\ell \neq m$ and $\ell = m$, where ℓ and m represent nodes of the network.

In order to obtain a recursion for $\mathbf{S}_{\ell m}(n)$, we must rewrite (2) in terms of the weight-error vector $\widetilde{\boldsymbol{\psi}}_k(n)$. Thus, subtracting both sides of (2) from $\mathbf{w}_{\mathrm{o}}(n)$ and replacing $e_k(n) = \mathbf{u}_k^T(n)\widetilde{\boldsymbol{\psi}}_k(n-1) + v_k(n)$, we obtain

$$\widetilde{\boldsymbol{\psi}}_k(n) = [\mathbf{I} - \widetilde{\mu}_k(n)\mathbf{u}_k(n)\mathbf{u}_k^T(n)]\,\widetilde{\boldsymbol{\psi}}_k(n-1) - \widetilde{\mu}_k(n)\mathbf{u}_k(n)v_k(n) + \mathbf{q}(n), \tag{53}$$

where $\mathbf{I}$ stands for the identity matrix of dimension M and

$$\widetilde{\mu}_k(n) \triangleq \frac{\mu_k}{\delta + \|\mathbf{u}_k(n)\|^2}. \tag{54}$$

Multiplying (53) with $k \leftarrow \ell$ by its transpose with $k \leftarrow m$, taking the expectations of both sides, and using the fact that $\mathrm{E}\{\widetilde{\boldsymbol{\psi}}_k(n-1)\mathbf{q}^T(n)\} = \mathbf{0}$ since the sequence $\{\mathbf{q}(n)\}$ is i.i.d. (Assumption **A1**), we obtain

$$\begin{aligned}
\mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell(n)\widetilde{\boldsymbol{\psi}}_m^T(n)\} \approx\ & \mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell(n-1)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\} \\
& \overbrace{-\mathrm{E}\{\widetilde{\mu}_m(n)\widetilde{\boldsymbol{\psi}}_\ell(n-1)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\mathbf{u}_m(n)\mathbf{u}_m^T(n)\}}^{\boldsymbol{\mathcal{A}}} \\
& \overbrace{-\mathrm{E}\{\widetilde{\mu}_\ell(n)\mathbf{u}_\ell(n)\mathbf{u}_\ell^T(n)\widetilde{\boldsymbol{\psi}}_\ell(n-1)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\}}^{\boldsymbol{\mathcal{B}}} \\
& \overbrace{+\mathrm{E}\{\widetilde{\mu}_\ell(n)\widetilde{\mu}_m(n)\mathbf{u}_\ell(n)\mathbf{u}_\ell^T(n)\widetilde{\boldsymbol{\psi}}_\ell(n-1)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\mathbf{u}_m(n)\mathbf{u}_m^T(n)\}}^{\boldsymbol{\mathcal{C}}} \\
& \overbrace{+\mathrm{E}\{\widetilde{\mu}_\ell(n)\widetilde{\mu}_m v_\ell(n)v_m(n)\mathbf{u}_\ell(n)\mathbf{u}_m^T(n)\}}^{\boldsymbol{\mathcal{D}}} \\
& \overbrace{-\mathrm{E}\{\widetilde{\mu}_m(n)v_m(n)\widetilde{\boldsymbol{\psi}}_\ell(n-1)\mathbf{u}_m^T(n)\}}^{\boldsymbol{\mathcal{E}}} \\
& \overbrace{-\mathrm{E}\{\widetilde{\mu}_\ell(n)v_\ell(n)\mathbf{u}_\ell(n)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\}}^{\boldsymbol{\mathcal{F}}} \\
& \overbrace{+\mathrm{E}\{\widetilde{\mu}_\ell(n)\widetilde{\mu}_m v_m(n)\mathbf{u}_\ell(n)\mathbf{u}_\ell^T(n)\widetilde{\boldsymbol{\psi}}_\ell(n-1)\mathbf{u}_m^T(n)\}}^{\boldsymbol{\mathcal{G}}} \\
& \overbrace{+\mathrm{E}\{\widetilde{\mu}_\ell(n)\widetilde{\mu}_m v_\ell(n)\mathbf{u}_\ell(n)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\mathbf{u}_m(n)\mathbf{u}_m^T(n)\}}^{\boldsymbol{\mathcal{H}}} \\
& +\mathrm{E}\{\mathbf{q}(n)\mathbf{q}^T(n)\}.
\end{aligned} \tag{55}$$

Now, using Assumptions **A2**, **A5** and **A6** (see below), we can evaluate the terms $\boldsymbol{\mathcal{A}}$-$\boldsymbol{\mathcal{H}}$ of (55):

$\boldsymbol{\mathcal{A}}$- Recalling that Assumption **A2** implies that $\widetilde{\boldsymbol{\psi}}_\ell(n-1)$ and $\widetilde{\boldsymbol{\psi}}_m(n-1)$ are independent

of $\mathbf{u}_m(n)$, the term $\boldsymbol{\mathcal{A}}$ can be approximated by

$$\begin{aligned}\boldsymbol{\mathcal{A}} &= \mathrm{E}\Big\{\mathrm{E}\Big\{\widetilde{\mu}_m(n)\widetilde{\boldsymbol{\psi}}_\ell(n-1)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\mathbf{u}_m(n)\mathbf{u}_m^T(n)|\mathbf{u}_m(n)\Big\}\Big\} \\ &\approx \mathrm{E}\left\{\widetilde{\mu}_m(n)\mathrm{E}\left\{\widetilde{\boldsymbol{\psi}}_\ell(n-1)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\right\}\mathbf{u}_m(n)\mathbf{u}_m^T(n)\right\} \\ &= \mathbf{S}_{\ell m}(n-1)\,\mathrm{E}\{\widetilde{\mu}_m(n)\mathbf{u}_m(n)\mathbf{u}_m^T(n)\}. \end{aligned} \tag{56}$$

We must obtain an approximation for

$$\mathrm{E}\{\widetilde{\mu}_m(n)\mathbf{u}_m(n)\mathbf{u}_m^T(n)\} = \mu_m\mathrm{E}\left\{\frac{\mathbf{u}_m(n)\mathbf{u}_m^T(n)}{\delta+\mathbf{u}_m^T(n)\mathbf{u}_m(n)}\right\}. \tag{57}$$

To arrive at a simple model, we also assume that

A5- The number of coefficients M is large enough for each element $\mathbf{u}_m(n)\mathbf{u}_m^T(n)$ in the numerator to be approximately independent from the denominator $\sum_{l=0}^{M-1}|u(n-l)|^2$. This is equivalent to applying the averaging principle of [38], since for large M, $\|\mathbf{u}_m(n)\|^2$ tends to vary slowly compared to the individual entries of $\mathbf{u}_m(n)\mathbf{u}_m^T(n)$.

A6- The regressors $\mathbf{u}_k(n)$, $k = 1, 2, \ldots, N$ are formed by a tapped-delay line with Gaussian entries and $\delta = 0$. This is a common assumption in the analysis of adaptive filters and leads to reasonable analytical results [27].

Under **A5** and **A6**, (57) can be approximated as [39, 40]

$$\mathrm{E}\{\widetilde{\mu}_m(n)\mathbf{u}_m(n)\mathbf{u}_m^T(n)\} \approx \frac{\mu_m}{\sigma_{u_m}^2(M-2)}\mathbf{R}_{mm} \tag{58}$$

and the term $\boldsymbol{\mathcal{A}}$ as

$$\boldsymbol{\mathcal{A}} \approx \frac{\mu_m}{\sigma_{u_m}^2(M-2)}\mathbf{S}_{\ell m}(n-1)\mathbf{R}_{mm} \tag{59}$$

where $\sigma_{u_m}^2$ is the variance of the input signal at node k.

$\boldsymbol{\mathcal{B}}$- Analogously, we obtain for $\boldsymbol{\mathcal{B}}$

$$\boldsymbol{\mathcal{B}} \approx \frac{\mu_\ell}{\sigma_{u_\ell}^2(M-2)}\mathbf{R}_{\ell\ell}\,\mathbf{S}_{\ell m}(n-1). \tag{60}$$

$\boldsymbol{\mathcal{C}}$- Under **A2**, it holds that

$$\boldsymbol{\mathcal{C}} = \mathrm{E}\,\{\widetilde{\mu}_\ell(n)\mathbf{u}_\ell(n)\mathbf{u}_\ell^T(n)\mathbf{S}_{\ell m}(n-1)\widetilde{\mu}_m(n)\mathbf{u}_m(n)\mathbf{u}_m^T(n)\} \tag{61}$$

Note that if the regression data of nodes ℓ and m are spatially independent (Assumption **A2**) and under **A5** and **A6**, (61) reduces to

$$\boldsymbol{\mathcal{C}} \approx \frac{\mu_\ell\mu_m}{\sigma_{u_\ell}^2\sigma_{u_m}^2(M-2)^2}\,\mathbf{R}_{\ell\ell}\mathbf{S}_{\ell m}(n-1)\mathbf{R}_{mm}. \tag{62}$$

For $m=\ell$, $\mathbf{R}_{mm}=\mathbf{R}_{\ell\ell}\neq\mathbf{0}$ and therefore, (61) reduces to (see, e.g., [39, 40])

$$\boldsymbol{\mathcal{C}}\approx\frac{\mu_\ell^2}{\sigma_{u_\ell}^4(M-2)(M-4)}\Big[2\mathbf{R}_{\ell\ell}\mathbf{S}_{\ell\ell}(n-1)\mathbf{R}_{\ell\ell}+\mathrm{Tr}(\mathbf{R}_{\ell\ell}\mathbf{S}_{\ell\ell}(n-1))\mathbf{R}_{\ell\ell}\Big]. \tag{63}$$

$\boldsymbol{\mathcal{D}}$- Under Assumption **A2** and for $\ell\neq m$, $\boldsymbol{\mathcal{D}}\approx\mathbf{0}$. On the other hand, for $m=\ell$, we get [15]

$$\boldsymbol{\mathcal{D}}\approx\frac{\mu_\ell^2}{\sigma_{u_\ell}^4(M-2)(M-4)}\sigma_{v_\ell}^2\mathbf{R}_{\ell\ell}. \tag{64}$$

$\boldsymbol{\mathcal{E}}$-$\boldsymbol{\mathcal{H}}$- Under Assumption **A2**, all the terms $\boldsymbol{\mathcal{E}}$ to $\boldsymbol{\mathcal{H}}$ are $M\times M$ null matrices.

From the previous results, (55) reduces to (34). Similarly, for $m=\ell$, we arrive at (35).

To obtain a recurrent expression for $\mathbf{X}_{\ell m}(n)\triangleq\mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell(n)\widetilde{\mathbf{w}}_m^T(n)\}$, we first add $\mathbf{q}(n)$ to both sides of (24) with $k\leftarrow m$ and transpose the resulting equation, which leads to

$$\widetilde{\mathbf{w}}_m^T(n)\approx c_{mm}(n)\widetilde{\boldsymbol{\psi}}_m^T(n-1)+\sum_{p\in\bar{\mathcal{N}}_m}c_{pm}(n)\widetilde{\mathbf{w}}_p^T(n-1)+\mathbf{q}^T(n). \tag{65}$$

Then, we multiply (65) by $\widetilde{\boldsymbol{\psi}}_\ell(n)$ from the left, using (53) with $k\leftarrow\ell$ to multiply the right-hand side. Taking expectations on both sides and using Assumptions **A1**-**A3**, we arrive at

$$\begin{aligned}\mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell(n)\widetilde{\mathbf{w}}_m^T(n)\}\approx\ &\mathrm{E}\{c_{mm}(n)\}\mathrm{E}\{[\mathbf{I}-\widetilde{\mu}_\ell(n)\mathbf{u}_\ell(n)\mathbf{u}_\ell(n)]\}\mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell(n-1)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\}\\&+\sum_{p\in\bar{\mathcal{N}}_m}\mathrm{E}\{c_{pm}(n)\}\mathrm{E}\{[\mathbf{I}-\widetilde{\mu}_\ell(n)\mathbf{u}_\ell(n)\mathbf{u}_\ell(n)]\}\mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell(n-1)\widetilde{\mathbf{w}}_p^T(n-1)\}+\mathbf{Q}.\end{aligned} \tag{66}$$

Under Assumptions **A5** and **A6** [see Eq. (58)], (66) reduces to

$$\begin{aligned}\mathbf{X}_{\ell m}(n)\approx\ &\mathrm{E}\{c_{mm}(n)\}\left[\mathbf{I}-\frac{\mu_\ell}{\sigma_{u_\ell}^2(M-2)}\mathbf{R}_{\ell\ell}\right]\mathbf{S}_{\ell m}(n-1)\\&+\left[\mathbf{I}-\frac{\mu_\ell}{\sigma_{u_\ell}^2(M-2)}\mathbf{R}_{\ell\ell}\right]\sum_{p\in\bar{\mathcal{N}}_m}\mathrm{E}\{c_{pm}(n)\}\mathbf{X}_{\ell p}(n-1)+\mathbf{Q}.\end{aligned} \tag{67}$$

Using the definition of $\overline{\mu}_\ell$ [Eq. (38)] in (67), we arrive at (36).

Finally, to obtain a recurrent expression for $\mathbf{W}_{\ell m}(n)\triangleq\mathrm{E}\{\widetilde{\mathbf{w}}_\ell(n)\widetilde{\mathbf{w}}_m^T(n)\}$, we multiply (24) with $k\leftarrow\ell$ by its transpose with $k\leftarrow m$ from the right and take the expectations of both sides. After some algebraic manipulations under Assumptions **A1** and **A3**, we

arrive at

$$\begin{aligned}\mathrm{E}\{\widetilde{\mathbf{w}}_\ell(n)\widetilde{\mathbf{w}}_m^T(n)\} - \mathbf{Q} \approx\ & \mathrm{E}\{c_{\ell\ell}(n)c_{mm}(n)\}\mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell(n-1)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\} \\ & + \sum_{p\in\mathcal{N}_\ell}\sum_{r\in\mathcal{N}_m} \mathrm{E}\{c_{p\ell}(n)c_{rm}(n)\}\mathrm{E}\{\widetilde{\mathbf{w}}_p(n-1)\widetilde{\mathbf{w}}_r^T(n-1)\} \\ & + \sum_{r\in\mathcal{N}_m} \mathrm{E}\{c_{\ell\ell}(n)c_{rm}(n)\}\mathrm{E}\{\widetilde{\boldsymbol{\psi}}_\ell(n-1)\widetilde{\mathbf{w}}_r^T(n-1)\} \\ & + \sum_{p\in\mathcal{N}_\ell} \mathrm{E}\{c_{p\ell}(n)c_{mm}(n)\}\mathrm{E}\{\widetilde{\mathbf{w}}_p(n-1)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\}. \end{aligned} \tag{68}$$

Noting that $\mathrm{E}\{\widetilde{\mathbf{w}}_p(n-1)\widetilde{\boldsymbol{\psi}}_m^T(n-1)\} = \mathbf{X}_{mp}^T(n-1)$, (68) can be rewritten as (37).

References

[1] J. Fernándes-Bes, L. Azpicueta-Ruiz, M. T. M. Silva, J. Arenas-García, A novel scheme for diffusion networks with least-squares adaptive combiners, in: Proc. of IEEE Workshop on Machine Learning for Signal Process., Santander, Spain, 2012.

[2] C. G. Lopes, A. H. Sayed, Incremental adaptive strategies over distributed networks, IEEE Trans. Signal Process. 55 (2007) 4064–4077.

[3] C. G. Lopes, A. H. Sayed, Diffusion least-mean squares over adaptive networks: formulation and performance analysis, IEEE Trans. Signal Process. 56 (2008) 3122–3136.

[4] I. D. Schizas, G. Mateos, G. B. Giannakis, Distributed LMS for consensus-based in network adaptive processing, IEEE Trans. Signal Process. 57 (6) (2009) 2365–2382.

[5] F. S. Cattivelli, A. H. Sayed, Diffusion LMS strategies for distributed estimation, IEEE Trans. Signal Process. 58 (3) (2010) 1035–1048.

[6] F. S. Cattivelli, A. H. Sayed, Distributed detection over adaptive networks using diffusion adaptation, IEEE Trans. Signal Process. 59 (5) (2011) 1917–1932.

[7] S. Chouvardas, K. Slavakis, S. Theodoridis, Adaptive robust distributed learning in diffusion sensor networks, IEEE Trans. Signal Process. 59 (10) (2011) 4692–4707.

[8] A. Khalili, M. Tinati, A. Rastegarnia, J. Chambers, Steady-state analysis of diffusion lms adaptive networks with noisy links, IEEE Trans. Signal Process. 60 (2) (2012) 974–979.

[9] N. Takahashi, I. Yamada, A. H. Sayed, Diffusion least-mean squares with adaptive combiners: Formulation and performance analysis, IEEE Trans. Signal Process. 58 (2010) 4795–4810.

[10] S.-Y. Tu, A. H. Sayed, Optimal combination rules for adaptation and learning over networks, in: Proc. of 4th IEEE Intl. Workshop on Computational Advances in Multi-Sensor Adaptive Process. (CAMSAP), San Juan, Puerto Rico, 2011, pp. 317–320.

[11] X. Zhao, A. H. Sayed, Performance limits for distributed estimation over LMS adaptive networks, IEEE Trans. Signal Process. 60 (10) (2012) 5107–5124.

[12] A. H. Sayed, Diffusion adaptation over networks, in: R. Chellapa, S. Theodoridis (Eds.), Academic Press Library in Signal Processing: array and statistical signal processing, Vol. 3, Academic Press, 2014, Ch. 9, pp. 323–456.

[13] S.-Y. Tu, A. H. Sayed, Mobile adaptive networks, IEEE Journal of Selected Topics in Signal Process. 5 (4) (2011) 649–664.

[14] P. Di Lorenzo, S. Barbarossa, A. H. Sayed, Bio-inspired decentralized radio access based on swarming mechanisms over adaptive networks, IEEE Trans. Signal Process. 61 (12) (2013) 3183–3197.

[15] A. H. Sayed, Adaptive Filters, John Wiley & Sons, NJ, 2008.

[16] R. Arroyo-Valles, S. Maleki, G. Leus, A censoring strategy for decentralized estimation in energy-constrained adaptive diffusion networks, in: Proc. of IEEE 14th Workshop on Signal Process. Advances in Wireless Commun. (SPAWC), Darmstadt, Germany, 2013, pp. 155-159.

[17] V. Krishnamurthy, Decentralized activation in sensor networks ï¿½- global games and adaptive filtering games, Digital Signal Process. 21 (2011) 638–647.

[18] C.-K. Yu, A. H. Sayed, A strategy for adjusting combination weights over adaptive networks, in: Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Process. (ICASSP), Vancouver, Canada, 2013, pp. 4579–4583.

[19] J. Arenas-García, M. Martínez-Ramón, A. Navia-Vázquez, A. R. Figueiras-Vidal, Plant identification via adaptive combination of transversal filters, Signal Process. 86 (2006) 2430–2438.

[20] J. Arenas-García, A. R. Figueiras-Vidal, A. H. Sayed, Mean-square performance of a convex combination of two adaptive filters, IEEE Trans. Signal Process. 54 (2006) 1078–1090.

[21] L. A. Azpicueta-Ruiz, A. R. Figueiras-Vidal, J. Arenas-García, A normalized adaptation scheme for the convex combination of two adaptive filters, in: Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process., Las Vegas, NV, 2008, pp. 3301–3304.

[22] M. T. M. Silva, V. H. Nascimento, Improving the tracking capability of adaptive filters via convex combination, IEEE Trans. Signal Process. 56 (2008) 3137–3149.

[23] V. H. Nascimento, M. T. M. Silva, J. Arenas-García, A low-cost implementation strategy for combination of adaptive filters, in: Proc. of IEEE Int. Conf. on Acoustics, Speech, and Signal Process. (ICASSP), Vancouver, Canada, 2013, pp. 5671–5675.

[24] N. J. Bershad, J. C. M. Bermudez, J.-Y. Tourneret, An affine combination of two LMS adaptive filters - transient mean-square analysis, IEEE Trans. Signal Process. 56 (2008) 1853–1864.

[25] L. A. Azpicueta-Ruiz, A. R. Figueiras-Vidal, J. Arenas-García, A new least squares adaptation scheme for the affine combination of two adaptive filters, in: IEEE Workshop on Machine Learning for Signal Process., Cancún, Mexico, 2008, pp. 327–332.

[26] L. A. Azpicueta-Ruiz, M. Zeller, A. R. Figueiras-Vidal, J. Arenas-García, Least-squares adaptation of affine combinations of multiple adaptive filters, in: Proc. of IEEE Intl. Symp. on Circuits and Systems, Paris, France, 2010, pp. 2976–2979.

[27] S. Haykin, Adaptive Filter Theory, 4th Edition, Prentice Hall, Upper Saddle River, 2001.

[28] M. T. M. Silva, V. H. Nascimento, J. Arenas-García, A transient analysis for the convex combination of two adaptive filters with transfer of coefficients, in: Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process., Dallas, TX, 2010, pp. 3842–3845.

[29] M. Niedźwieck, Identification of Time-varying Processes, Wiley, 2000.

[30] T. Sadiki, M. Triki, D. T. M. Slock, Window optimization issues in recursive least-squares adaptive filtering and tracking, in: Proc. 38th Asilomar Conf. Signals, Systems, and Computers, Vol. 1, Pacific Groove, CA, 2004, pp. 940–944.

[31] M. Niedźwiecki, On tracking characteristics of weighted least squares estimators applied to nonstationary system identification, IEEE Trans. Autom. Control 33 (1) (1988) 96–98.

[32] M. S. E. Abadi, F. Moradiani, A unified approach to tracking performance analysis of the selective partial update adaptive filter algorithms in nonstationary environment, Digital Signal Process. 23 (2013) 817–830.

[33] J. E. Mazo, On the independent theory of equalizer convergence, Bell Syst. Tech. J. 58 (1979) 963–993.

[34] J. Minkoff, Comment: on the unnecessary assumption of statistical independence between reference signal and filter weights in feedforward adaptive systems, IEEE Trans. Signal Process. 49 (2001) 1109.

[35] R. Candido, M. T. M. Silva, V. H. Nascimento, Transient and steady-state analysis of the affine combination of two adaptive filters, IEEE Trans. Signal Process. 58 (8) (2010) 4064–4078.

[36] S. S. Kozat, A. T. Erdogan, A. C. Singer, A. H. Sayed, Steady-state MSE performance analysis of mixture approaches to adaptive filtering, IEEE Trans. Signal Process. 58 (8) (2010) 4050–4063.

[37] M. Lázaro-Gredilla, L. A. Azpicueta-Ruiz, A. R. Figueiras-Vidal, J. Arenas-García, Adaptively biasing the weights of adaptive filters, IEEE Trans. Signal Process. 58 (2010) 3890–3895.

[38] C. Samson, V. U. Reddy, Fixed poit error analysis of the normalized ladder algorithms, IEEE Trans. Acoust., Speech, Signal Process. 31 (1983) 1177–1191.

[39] M. C. Costa, J. C. M. Bermudez, An improved model for the normalized LMS algorithm with Gaussian inputs and large number of coefficients, in: Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Process., Vol. II, Orlando, FL, 2002, pp. 1385–1388.

[40] N. J. Bershad, J. C. M. Bermudez, Mean-square stability of the Normalized Least-Mean Fourth algorithm for white Gaussian inputs, Digital Signal Process. 21 (2011) 694–700.