Affine Projection Algorithm Over Acoustic Sensor Networks for Active Noise Control

Miguel Ferrer, *Member, IEEE*, Maria de Diego, *Senior Member, IEEE*, Gema Piñero, *Senior Member, IEEE*, and Alberto Gonzalez, *Senior Member, IEEE*

***Abstract*—Acoustic sensor networks (ASNs) are an effective solution to implement active noise control (ANC) systems by using distributed adaptive algorithms. On one hand, ASNs provide scalable systems where the signal processing load is distributed among the network nodes. On the other hand, their noise reduction performance is comparable to that of their respective centralized processing systems. In this sense, the distributed multiple error filtered-x least mean squares (DMEFxLMS) adaptive algorithm has shown to obtain the same performance than its centralized counterpart as long as there are no communications constraints in the underlying ASN. Regarding affine projection (AP) adaptive algorithms, some distributed approaches that are approximated versions of the multichannel filtered-x affine projection (MFxAP) algorithm have been previously proposed. These AP algorithms can efficiently share the processing load among the nodes, but at the expense of worsening their convergence properties. In this paper we develop the exact distributed multichannel filtered-x AP (EFxAP) algorithm, which obtains the same solution as that of the MFxAP algorithm as long as there are no communications constraints in the underlying ASN. In the EFxAP algorithm each node can compute a part or the entire inverse matrix needed by the centralized MFxAP algorithm. Thus, we propose three different strategies that obtain significant computational saving: 1) Gauss Elimination, 2) block LU factorization, and 3) matrix inversion lemma. As a result, each node computes only between 25%–60% of the number of multiplications required by the direct inversion of the matrix. Regarding the performance in transient and steady states, the EFxAP exhibits the fastest convergence and the highest noise level reduction for any size of the acoustic network and any projection order of the AP algorithm compared to the DMEFxLMS and two previously reported distributed AP algorithms.**

***Index Terms*—Active noise control, acoustic sensor networks, affine projection algorithm, distributed algorithms, adaptive filters.**

I. Introduction

THE use of acoustic sensor networks (ASNs) [1] as an alternative to fixed multi-channel sound systems is an emerging topic that has attracted significant attention from the signal processing community over the past years due to the scalability and versatility of the ASNs. One of the benefits of ASNs is their ability to obtain acoustical information from the environment to perform different tasks such as source monitoring [2]–[4], localization [5], [6] or detection [7] in a distributed way. Examples of a large variety of applications as well as the inherent challenges that need to be addressed can be found in the recent literature, such as multiple sound source location [8]–[10], speech enhancement [11], [12] or blind synchronization [13] among others. It is generally considered that the nodes of the network are capable of monitoring the environment and therefore they are equipped with one or more microphones and a processor with limited processing and networking capabilities. However, in the particular case of sound field control applications, and more specifically for Active Noise Control (ANC), the acoustic nodes must also be equipped with an actuator in order to interact with their surrounding zone and emit the signals needed to control the sound field [14]–[17]. Moreover, every node should communicate with the rest of the nodes to exchange information such that they can manage and process their own signals in a proper way. The aim is to exchange updated system information, but no signals, so the computational burden is distributed throughout the nodes by means of ad-hoc distributed adaptive algorithms [14], [15].

It is well known that ANC systems are devoted to reduce the undesired, or primary, noise by the addition of a secondary sound specifically designed to cancel the first one. Feedforward adaptive controllers estimate the signals that will feed the secondary sources based on a reference noise, which is assumed to be available at the controller. Assuming as well that the acoustic path between the secondary source and the error microphone is known, the adaptive algorithm tries to compensate the effect of the secondary path by means of the widely used filtered-x structure [18].

Regarding the adaptive algorithms proposed for a multichannel ANC system comprised of J loudspeakers and K microphones, solutions for a single centralized controller can be found in the literature for the particular case of the affine projection (AP) algorithm [19]. The different AP methods proposed in [20]–[26] improve the convergence performance of the well-known least-mean-square (LMS) algorithm [27] at the expense of increasing the computational complexity. AP algorithms using the filtered-x scheme, known as filtered-x AP

Manuscript received March 27, 2020; revised July 31, 2020 and November 17, 2020; accepted November 19, 2020. Date of publication December 8, 2020; date of current version December 28, 2020. This work was supported by EU together with Spanish Government through RTI2018-098085-B-C41 (MINECO/FEDER) and Generalitat Valenciana through PROMETEO/2019/109. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Stefania Cecchi. *(Corresponding author: Gema Pinero.)*

The authors are with the Institute of Telecommunications and Multimedia Applications (iTEAM), Universitat Politecnica de Valencia (UPV), 46022 Valencia, Spain (e-mail: mferrer@dcom.upv.es; mdediego@dcom.upv.es; gpinyero@iteam.upv.es; agonzal@dcom.upv.es).

Digital Object Identifier 10.1109/TASLP.2020.3042590

(FxAP) algorithms [28]–[34], have shown to be robust and stable and have been proposed as a suitable alternative to the filtered-x LMS (FxLMS) algorithm [18].

Consider now the same multichannel ANC system comprised of J loudspeakers and K microphones but supported by an acoustic sensor network with one processing unit per node. ASNs are usually designed to waste as less energy as possible and their computational capacity per node is also limited. Several adaptive algorithms have been proposed for the implementation of ANC systems over ASNs. Two decentralized ANC systems were introduced in [35], [36], where the nodes shared the computational load of an equivalent centralized system, but did not collaborate in order to reach the same solution as in the centralized case, thus compromising the stability of the ANC system when the transducers were acoustically coupled. On the other hand, [14], [37], [38] present decentralized ANC schemes that include cooperation among subsystems or nodes. The control model in [37] compensates for the interference of the primary sound at each location of interest by exchanging run-time data amongst the control units. In [38] the control method shapes the eigenvalues of a matrix that models the two-channel secondary paths for each frequency bin. In [14], a distributed multiple error filtered-x LMS (DMEFxLMS) algorithm over ASNs based on incremental communication among the nodes [39] was introduced. The incremental communication is carried out over a ring network where the updating of the adaptive algorithm is performed sequentially: Node k receives some data from node $k-1$, partially updates the global adaptive filter with the help of the received data and its local information, and passes some new data to node $k+1$. Assuming perfect network synchronization, the DMEFxLMS in [14] obtained the same performance as its corresponding centralized multichannel system.

Recently, and motivated by the good trade-off between convergence speed and computational cost of the AP algorithms, two approaches that address the distribution of the AP processing over ASNs have been presented [15], [16]. However, they are approximated distributed versions of the centralized multichannel FxAP (MFxAP) algorithm [31]–[34] and, from their analysis, it is not assured that they can obtain the same performance as the MFxAP.

In this paper, we introduce an exact multichannel FxAP (EFxAP) algorithm that gives the same solution as the MFxAP, but it can be computed in a distributed way over ASNs. As we will show in Section III, to evolve the centralized MFxAP algorithm into the distributed EFxAP algorithm could be done somehow straightforwardly by computing at each node the same inverse matrix that the MFxAP computes once. However, it would be very inefficient since most of the computational cost carried out once by the MFxAP would have to be computed as many times as the number of nodes in the network. Therefore this straightforward solution would be inefficient for most practical cases where the distributed processors at the nodes have a limited computational capacity [40].

In this work we propose three robust strategies that reduce the computational requirements of the EFxAP at each node. These approaches are the Gauss elimination (GE), the block LU factorization, and the matrix inversion lemma methods [41], which efficiently decrease the computational cost at each node but obtain the same good performance as the centralized MFxAP algorithm. Simulations using real acoustic channel responses have been carried out in order to assess the performance of the proposed EFxAP over ASNs with incremental communication. Comparisons with previous distributed LMS-type and AP-type adaptive algorithms have been provided showing a faster convergence and a higher noise reduction for all the scenarios. The effect of network latency and computation time of the different strategies on the EFxAP performance has also been discussed, concluding that EFxAP can bear a latency of a few sampling times at the cost of degrading its performance, similar to the observed effect on other distributed adaptive algorithms [14].

The remainder of the paper is structured as follows. In Section II we formulate the multichannel sound control problem and the MFxAP algorithm [31] is also motivated and described. Section III presents the EFxAP algorithm over an acoustically coupled sensor network with incremental communication among the nodes. Three different strategies to improve the computational efficiency of EFxAP are proposed in Section IV whereas Section V studies their corresponding computational costs. Section VI evaluates the performance of the EFxAP compared to the DMEFxLMS [14] and the two approximated distributed filtered-x AP algorithms in [15], [16]. The main conclusions are summarized in Section VII.

The following notation is used throughout the paper: boldface upper-case letters denote matrices (e.g. $\mathbf{A}$), boldface lower-case letters denote vectors (e.g., $\mathbf{a}$), and italics denote scalars, (e.g. a or A). The Euclidean norm is denoted by $\|\cdot\|$, $(\cdot)^T$ stands for matrix or vector transpose, $\mathbf{I}_a$ is an $a \times a$ identity matrix and $\mathbf{0}_{a\times b}$ is an $a \times b$ matrix of 0's respectively. The most frequent symbols, signals, vectors and matrices used throughout the paper can also be found in Table I.

II. Multichannel Sound Control Problem

Let us consider a generic linear multichannel ANC system comprised of J secondary sources (loudspeakers) and K error sensors (microphones). In order to cancel the undesired noise signal $d_k(n)$ originated from one or several noise sources captured by sensor k, the following equation must be fulfilled at the kth microphone, $k = 1, \ldots, K$,

$$d_k(n) = \sum_{j=1}^{J} [-y_j(n) * h_{jk}(n)], \tag{1}$$

where $*$ denotes the discrete linear convolution, $h_{jk}(n)$ is the impulse response of the M-length FIR filter that models the acoustic channel between the jth loudspeaker and the kth microphone, and $y_j(n)$ is the signal generated by the jth loudspeaker. The impulse response $h_{jk}(n)$ can be written in vector form as $\mathbf{h}_{jk} = [h_{jk}(0), h_{jk}(1), \ldots, h_{jk}(M-1)]^T$.

The output signal of the jth loudspeaker, $j = 1, \ldots, J$, is obtained as

$$y_j(n) = \mathbf{w}_j^T(n)\mathbf{x}(n)\,, \tag{2}$$

where $\mathbf{x}(n)$ is an $L \times 1$ vector formed by the most recent L samples of the reference signal $x(n)$ at time n. It is assumed

TABLE I
FREQUENTLY-USED SYMBOLS, SIGNALS, VECTORS, AND MATRICES USED THROUGHOUT THE PAPER

Notation	Definition
J	Number of actuators or loudspeakers
K	Number of error sensors or microphones
L	Length of the adaptive filters
N	Projection order of the AP algorithm
$d_k(n)$	Undesired signal at the kth microphone
$x(n)$	Reference signal
$y_j(n)$	Signal emitted by the jth actuator
$\mathbf{w}_j(n)$	Vector with the adaptive filter coefficients at time n linking reference signal $x(n)$ and $y_j(n)$
$\widehat{\mathbf{h}}_{jk}$	Vector with the coefficients of the FIR filter that models the acoustic channel between the jth actuator and the kth microphone
$\mathbf{v}_{jk}(n)$	Vector with the last L samples of reference signal $x(n)$ filtered by acoustic channel $\widehat{\mathbf{h}}_{jk}$. Defined in (5)
$\mathbf{V}_{jk}(n)$	Matrix formed by vectors $\mathbf{v}_{jk}$ from time $n-N+1$ to time n in reverse order. Defined in (10)
$\mathbf{e}_k(n)$	Vector with the last N samples of the error signal at the kth microphone

that $x(n)$ is correlated with the unwanted noise captured by the sensors, $d_k(n)$. The $L \times 1$ vector $\mathbf{w}_j(n)$ contains the coefficients of the adaptive filter associated to the jth actuator. Consequently, we can form the $JL \times 1$ vector $\mathbf{w}(n)$ by stacking all the filters $\mathbf{w}_j(n)$ in a single column vector:

$$\mathbf{w}(n) = \left[\, \mathbf{w}_1^T(n)\ \mathbf{w}_2^T(n)\ \ldots\ \mathbf{w}_J^T(n) \,\right]^T . \tag{3}$$

Consider now that a certain adaptive algorithm obtains a solution of (1)–(2) denoted by $\mathbf{w}_0 = \left[\, \mathbf{w}_{01}^T\ \mathbf{w}_{02}^T\ \ldots\ \mathbf{w}_{0J}^T \,\right]^T$, then the undesired signal at the kth microphone (1) can be expressed as

$$d_k(n) = \sum_{j=1}^{J} \left[-\mathbf{v}_{jk}^T(n)\mathbf{w}_{0j} \right] , \tag{4}$$

where $\mathbf{v}_{jk}(n)$ denotes an $L \times 1$ vector obtained by filtering the most-recent samples of the reference signal through:

$$\mathbf{v}_{jk}(n) = \mathbf{X}(n)\widehat{\mathbf{h}}_{jk} , \tag{5}$$

where the $L \times M$ Toeplitz matrix $\mathbf{X}(n)$ is defined as

$$\mathbf{X}(n) = \left[\, \mathbf{x}(n)\ \mathbf{x}(n-1)\ \cdots\ \mathbf{x}(n-M+1) \,\right], \tag{6}$$

and $\widehat{\mathbf{h}}_{jk}$ is an accurate estimation of the acoustic channel $\mathbf{h}_{jk}$ between jth loudspeaker and kth microphone defined in (1), that is, $\widehat{\mathbf{h}}_{jk} \approx \mathbf{h}_{jk}$.

Notice that the perfect filtering equation (4) could be fulfilled at the K microphones by more than one solution $\mathbf{w}_0$ since the number of equations is K, the number of unknowns is JL, and the system is commonly underdetermined ($K < JL$). Therefore, the perfect filtering equation (4) can be extended up to $(N-1)$ past samples of the desired signals,

$$d_k(n-i) = \sum_{j=1}^{J} [-\mathbf{v}_{jk}^T(n-i)\mathbf{w}_{0j}], \quad 0 \le i < N , \tag{7}$$

and can be fulfilled by more than one filter vector $\mathbf{w}_0$ as long as $KN < JL$, being N the projection order.

We can express (7) in compact form for $N \ge 1$ as

$$\mathbf{d}(n) = -\mathbf{V}^T(n)\mathbf{w}_0, \tag{8}$$

where $\mathbf{d}(n) = \left[\, \mathbf{d}_1^T(n),\ \mathbf{d}_2^T(n),\ \ldots,\ \mathbf{d}_K^T(n) \,\right]^T$ is a $KN \times 1$ vector formed by blocks $\mathbf{d}_k(n)$ containing the last N samples of the undesired signal $d_k(n)$ at the kth microphone, and $\mathbf{V}(n)$ is a $JL \times KN$ matrix defined as

$$\mathbf{V}(n) = \begin{bmatrix} \mathbf{V}_{11}(n) & \mathbf{V}_{12}(n) & \cdots & \mathbf{V}_{1K}(n) \\ \mathbf{V}_{21}(n) & \mathbf{V}_{22}(n) & \cdots & \mathbf{V}_{2K}(n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{V}_{J1}(n) & \mathbf{V}_{J2}(n) & \cdots & \mathbf{V}_{JK}(n) \end{bmatrix}, \tag{9}$$

where $\mathbf{V}_{jk}(n)$ is an $L \times N$ matrix obtained by filtering the reference signal $x(n)$ through the secondary acoustic path $\widehat{\mathbf{h}}_{jk}$ such that

$$\mathbf{V}_{jk}(n) = \Big[\mathbf{v}_{jk}(n), \mathbf{v}_{jk}(n-1), \ldots, \mathbf{v}_{jk}(n-N+1) \Big] . \tag{10}$$

A. Centralized Multichannel Filtered-x AP Solution

The adaptive filter $\mathbf{w}(n)$ defined in (3) can be calculated by solving a constrained optimization problem derived from the *minimum perturbation principle* [32], [42], and whose solution was presented in [29] as an extension of the single channel to the multiple channel filtered-x AP (MFxAP):

$$\min_{\mathbf{w}(n)} \|\mathbf{w}(n) - \mathbf{w}(n-1)\|^2, \tag{11}$$

subject to the perfect filtering equation system (8)

$$\mathbf{d}(n) + \mathbf{V}^T(n)\mathbf{w}(n) = \mathbf{0}_{KN\times 1}. \tag{12}$$

Using the method of Lagrange multipliers [42] to solve (12), the cost criterion becomes

$$J(n) = \|\Delta\mathbf{w}(n)\|^2 + [\mathbf{d}(n) + \mathbf{V}^T(n)\mathbf{w}(n)]^T\boldsymbol{\lambda}(n), \tag{13}$$

where $\boldsymbol{\lambda}(n)$ is the $KN \times 1$ vector that comprises the KN Lagrange multipliers corresponding to the KN constraints at the K microphones and $\Delta\mathbf{w}(n) = \mathbf{w}(n) - \mathbf{w}(n-1)$.

To solve (13), we calculate the gradient of $J(n)$ with respect to the weight vector $\mathbf{w}(n)$ as

$$\nabla_w J(n) = \frac{\partial J(n)}{\partial \mathbf{w}(n)} = 2\Delta\mathbf{w}(n) + \mathbf{V}(n)\boldsymbol{\lambda}(n), \tag{14}$$

and the minimum of $J(n)$ is obtained by

$$\mathbf{w}(n) = \mathbf{w}(n-1) - \frac{1}{2}\mathbf{V}(n)\boldsymbol{\lambda}(n). \tag{15}$$

Substituting (15) in the constraint relation (12) yields

$$\mathbf{d}(n) = -\mathbf{V}^T(n)\mathbf{w}(n-1) + \frac{1}{2}[\mathbf{V}^T(n)\mathbf{V}(n)]\boldsymbol{\lambda}(n). \quad (16)$$

Thus, solving (16) for the Lagrange vector, we obtain

$$\boldsymbol{\lambda}(n) = 2[\mathbf{V}^T(n)\mathbf{V}(n)]^{-1}\mathbf{e}(n), \quad (17)$$

where

$$\mathbf{e}(n) = \mathbf{d}(n) + \mathbf{V}^T(n)\mathbf{w}(n-1) \quad (18)$$

is the *a priori* error vector of length KN expressed as $\mathbf{e}(n) = [\mathbf{e}_1^T(n), \mathbf{e}_2^T(n), \ldots, \mathbf{e}_K^T(n)]^T$, where each $N \times 1$ vector $\mathbf{e}_k(n)$ corresponds to the *a priori* error signal recorded by the kth microphone. Finally, substituting (17) into (15) and introducing both an step size parameter μ to control the convergence speed and a regularization factor δ, the adaptation rule of the MFxAP algorithm is given by

$$\mathbf{w}(n) = \mathbf{w}(n-1) - \mu\mathbf{V}(n)[\mathbf{V}^T(n)\mathbf{V}(n) + \delta\mathbf{I}_{KN}]^{-1}\mathbf{e}(n). \quad (19)$$

The filter updating equation in (19) can be rewritten as

$$\mathbf{w}(n) = \mathbf{w}(n-1) - \mu\mathbf{V}(n)\mathbf{B}(n)\mathbf{e}(n)\,, \quad (20)$$

where $\mathbf{B}(n) = [\mathbf{V}^T(n)\mathbf{V}(n) + \delta\mathbf{I}_{KN}]^{-1}$.

III. Derivation of the Exact Distributed Filtered-x AP Algorithm

In this section we extend the MFxAP algorithm for distributed ANC over ASNs under the assumption of no rate or latency constraints in the network communication system. This "ideal" case can be assumed if all the communication and processing times required for one iteration of the adaptive algorithm are shorter than the sampling time. A discussion on the case that this condition is not fulfilled is provided in Section VI.

For the sake of clarity, we will consider from now on an homogeneous network of single-channel acoustic nodes, that is, each acoustic node is formed by one microphone, one loudspeaker and one unit with processing and communication capabilities [14]. Therefore, we assume an acoustic network of K single-channel nodes with ring topology and incremental communication that supports the ANC system composed of K error sensors and K secondary sources ($J = K$), as shown in Fig. 1 [16]. The aim is to distribute the computational burden among the different nodes so that each node performs its corresponding processing relying only on its local signal and on some data from the rest of the nodes, avoiding any signal exchange between them. We also assume that the reference signal $x(n)$ is available at each node, as it is commonly assumed in the literature of ANC over acoustic networks [17], [38], [43].

In order to properly distribute the processing among the network nodes while ensuring the exactly same solution as the one provided by the MFxAP algorithm, the full error vector $\mathbf{e}(n)$ in (20) must be decoupled for each microphone. Let us define matrices

$$\mathbf{A}(n) = \mathbf{V}^T(n)\mathbf{V}(n), \quad (21)$$

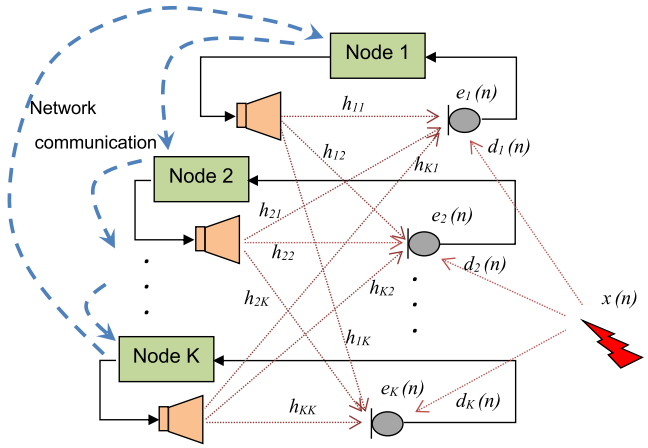

Fig. 1. Acoustic network for active noise control using ring topology and incremental communication.

and

$$\mathbf{B}(n) = \mathbf{A}^{-1}(n) = (\mathbf{V}^T(n)\mathbf{V}(n))^{-1}, \quad (22)$$

where we have omitted the regularization term $\delta\mathbf{I}_{KN}$ in the definition of $\mathbf{B}(n)$ for the sake of clarity. This term is usually included in affine projection methods in order to avoid the inversion of ill-conditioned matrices [44]. In this sense, we have omitted the explicit reference to the regularization term in (22), but we consider that a value of δ can be added to the elements of the diagonal of matrix $\mathbf{A}(n)$ in (21). Both matrices $\mathbf{A}(n)$ and $\mathbf{B}(n)$ have dimensions $KN \times KN$ and, since their structure is block-wise, they can be split into $K \times K$ submatrices of $N \times N$ size such that

$$\mathbf{A}(n) = \begin{bmatrix} \mathbf{A}_{11}(n) & \mathbf{A}_{12}(n) & \cdots & \mathbf{A}_{1K}(n) \\ \mathbf{A}_{21}(n) & \mathbf{A}_{22}(n) & \cdots & \mathbf{A}_{2K}(n) \\ & \ddots & & \\ \mathbf{A}_{K1}(n) & \mathbf{A}_{K2}(n) & \cdots & \mathbf{A}_{KK}(n) \end{bmatrix}, \quad (23)$$

and

$$\mathbf{B}(n) = \begin{bmatrix} \mathbf{B}_{11}(n) & \mathbf{B}_{12}(n) & \cdots & \mathbf{B}_{1K}(n) \\ \mathbf{B}_{21}(n) & \mathbf{B}_{22}(n) & \cdots & \mathbf{B}_{2K}(n) \\ & \ddots & & \\ \mathbf{B}_{K1}(n) & \mathbf{B}_{K2}(n) & \cdots & \mathbf{B}_{KK}(n) \end{bmatrix}. \quad (24)$$

where the diagonal elements of matrices $\mathbf{A}_{kk}(n)$ in (23) include the regularization term such that $a_{kk,jj} + \delta, \forall j, \forall k$. The proposed EFxAP can be derived from (20) decoupling the terms involving their local signals as

$$\mathbf{w}(n) = \mathbf{w}(n-1) - \mu\mathbf{V}(n)\sum_{k=1}^{K}\mathbf{B}_k(n)\mathbf{e}_k(n), \quad (25)$$

where the $KN \times N$ matrix $\mathbf{B}_k(n)$ corresponds to the block-wise kth column of $\mathbf{B}(n)$ in (24):

$$\mathbf{B}(n) = [\mathbf{B}_1(n) \quad \mathbf{B}_2(n) \quad \ldots \quad \mathbf{B}_K(n)]. \quad (26)$$

To calculate the adaptive filter coefficients $\mathbf{w}(n)$ in a distributed way over a ring topology with incremental communication, the kth node must add its own term to the summation in (25). Let us assume that at time n the first node of Fig. 1 has available the updated global vector obtained at time $n-1$, $\mathbf{w}(n-1)$. Then the first node computes

$$\mathbf{w}^{(1)}(n) = \mathbf{w}(n-1) - \mu \mathbf{V}(n)\mathbf{B}_1(n)\mathbf{e}_1(n), \tag{27}$$

where $\mathbf{w}^{(1)}(n)$ can be considered as the local version of the $KL \times 1$ global vector $\mathbf{w}(n)$ at node $k=1$. Once $\mathbf{w}^{(1)}(n)$ is calculated at the first node, it is transmitted to the second node which computes its local version as:

$$\mathbf{w}^{(2)}(n) = \mathbf{w}^{(1)}(n) - \mu \mathbf{V}(n)\mathbf{B}_2(n)\mathbf{e}_2(n). \tag{28}$$

By simple induction, the updating equation of the global adaptive filter at the kth node can be expressed as

$$\mathbf{w}^{(k)}(n) = \mathbf{w}^{(k-1)}(n) - \mu \mathbf{V}(n)\mathbf{B}_k(n)\mathbf{e}_k(n). \tag{29}$$

Once the last node is updated, the exactly same solution $\mathbf{w}(n)$ of the centralized MFxAP in (20) is obtained since

$$\begin{aligned}\mathbf{w}^{(K)}(n) &= \mathbf{w}^{(K-1)}(n) - \mu \mathbf{V}(n)\mathbf{B}_K(n)\mathbf{e}_K(n) \\ &= \mathbf{w}^{(K-2)}(n) - \mu \mathbf{V}(n)\mathbf{B}_{K-1}(n)\mathbf{e}_{K-1}(n) \\ &\quad - \mu \mathbf{V}(n)\mathbf{B}_K(n)\mathbf{e}_K(n) = \cdots \\ &= \mathbf{w}(n-1) - \mu \mathbf{V}(n) \sum_{k=1}^{K} \mathbf{B}_k(n)\mathbf{e}_k(n),\end{aligned} \tag{30}$$

that is, $\mathbf{w}^{(K)}(n) = \mathbf{w}(n)$. Then, the final value of $\mathbf{w}(n)$ will be shared with the rest of the nodes throughout a second communication round through the network, or through any other communication procedure established by the acoustic network. As said before, assuming no rate or latency constraints, the convergence properties of the EFxAP are equal to that of the centralized MFxAP algorithm since their global solution $\mathbf{w}(n)$ at time n is coincident. On the other hand, although all the nodes have to cooperate to calculate the global solution $\mathbf{w}(n)$, node k will only need the $L \times 1$ block $\mathbf{w}_k(n)$ from (3) to calculate its local output signal $y_k(n)$ as in (2).

Regarding the local calculation of the second term of (29), it should be noted that the whole matrix $\mathbf{V}(n)$ has to be available at the kth node. In this sense, since a setup stage of the acoustic network must be provided to estimate the acoustic paths $\widehat{\mathbf{h}}_{jk}$ for $j,k = 1,\ldots,K$, once the estimation process is finished, the network could spread the estimated acoustic channels to the rest of the nodes. If all the acoustic paths $\widehat{\mathbf{h}}_{jk}$ are available at each node, matrix $\mathbf{V}(n)$ in (9) can be locally computed by means of $\mathbf{v}_{jk}(n)$ in (5) since the reference signal $x(n)$ is also available at each node.

The second term in (29) involves the computation of vector $\mathbf{B}_k(n)\mathbf{e}_k(n)$ where $\mathbf{B}_k(n)$ is the block-wise kth column in (26) and $\mathbf{B}(n) = \mathbf{A}^{-1}(n)$ as defined in (22). Although matrix $\mathbf{A}(n)$ can be computed at each node as in (21), its direct inversion would cost $O((KN)^3)$ operations. Therefore, we present in the next section three strategies to compute $\mathbf{B}_k(n)\mathbf{e}_k(n)$ that avoid the direct inversion of $\mathbf{A}(n)$.

As a final remark, we want to notice that matrix $\mathbf{A}(n)$ can also be computed in a distributed way since $\mathbf{A}(n) = \sum_{j=1}^{J} \mathbf{V}_j^T(n)\mathbf{V}_j(n)$ being $\mathbf{V}_j(n)$ the block-wise jth row of $\mathbf{V}(n)$ in (9). In this case, the acoustic network should also provide an extra round to compute and distribute the whole matrix $\mathbf{A}(n)$ before computing the updating of the filter (29). On the other hand, each product $\mathbf{V}_{jk}^T(n)\mathbf{V}_{jp}(n)$ is a symmetric Toeplitz matrix, which provides additional computational saving in the calculation of $\mathbf{A}(n)$.

IV. Strategies to Distribute the Processing

In this section, we propose three methods to calculate the updating equation at node k given by (29). Once the adaptive filter coefficients $\mathbf{w}^{(k-1)}(n)$ arrive from the previous node, the local processor can compute matrix $\mathbf{V}(n)$ based on the most recent samples of reference signal $x(n)$, and it can also update its error signal $\mathbf{e}_k(n)$. To compute the $KN \times N$ block-wise column $\mathbf{B}_k(n)$, we propose in the following three strategies, which significantly reduce the number of required operations per node. To simplify the notation, the dependence on the discrete-time index n has been removed for the first and second strategy. As said before, we have assumed for all the strategies that each node can compute matrix $\mathbf{A}(n)$ from the knowledge of matrix $\mathbf{V}(n)$.

A. S1: Strategy Based on Gaussian Elimination

Considering the particular structure of matrices $\mathbf{A}$ and $\mathbf{B}$ given by (23) and (24) respectively, and that $\mathbf{AB} = \mathbf{I}_{KN}$, $\mathbf{B}_k$ can be obtained solving

$$\mathbf{AB}_k = \mathbf{I'}_k, \quad k = 1,\ldots,K, \tag{31}$$

where $\mathbf{I'}_k$ is a $KN \times N$ matrix of zeros except for the kth block of size $N \times N$ that contains the identity matrix, $\mathbf{I}_N$.

Without loss of generality, we apply in the following the Gaussian elimination (GE) method for block matrices [41] to calculate (31) for the particular case of $k=1$:

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \cdots & \mathbf{A}_{1K} \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \cdots & \mathbf{A}_{2K} \\ & \ddots & & \\ \mathbf{A}_{K1} & \mathbf{A}_{K2} & \cdots & \mathbf{A}_{KK} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{11} \\ \mathbf{B}_{21} \\ \vdots \\ \mathbf{B}_{K1} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_N \\ \mathbf{0}_{N\times N} \\ \vdots \\ \mathbf{0}_{N\times N} \end{bmatrix}. \tag{32}$$

The aim of GE is to reduce the full equation system of (32) to a lower triangular form. For this purpose, we consider matrix $\mathbf{A}$ formed by K block-wise rows of K submatrices of dimensions $N \times N$ and the GE operations will be applied on symmetric matrices $\mathbf{A}_{ij}$ on a block-wise rows basis. Let us call the initial matrix $\mathbf{A}^{(K)}$ such that $\mathbf{A}^{(K)} = \mathbf{A}$, and denote by $\mathbf{A}_{ij}^{(K)}$ their ij-th submatrices. As we want to obtain a lower triangular reduction of $\mathbf{A}$, in the first iteration we take submatrix $\mathbf{A}_{KK}^{(K)}$ as the pivot element. Therefore, the first iteration computes the ij-th components of the reduced matrix as

$$\mathbf{A}_{ij}^{(K-1)} = \mathbf{A}_{ij}^{(K)} - \mathbf{A}_{iK}^{(K)}(\mathbf{A}_{KK}^{(K)})^{-1}\mathbf{A}_{Kj}^{(K)}, \tag{33}$$

with $1 \leq i, j \leq K-1$. After the first iteration, the last block-wise column of $\mathbf{A}^{(K-1)}$ is formed by $K-1$ matrices $\mathbf{A}_{iK}^{(K-1)} = \mathbf{0}_{N\times N}$, $1 \leq i \leq K-1$, and $\mathbf{A}_{KK}^{(K-1)} = \mathbf{I}_N$.

Further reduction is carried out in consecutive iterations applying (33) such that

$$\mathbf{A}_{ij}^{(k-1)} = \mathbf{A}_{ij}^{(k)} - \mathbf{A}_{ik}^{(k)}(\mathbf{A}_{kk}^{(k)})^{-1}\mathbf{A}_{kj}^{(k)}, \tag{34}$$

from $k = K$ to $k = 2$ and with $1 \leq i, j \leq k-1$ at each iteration. After the last iteration, the only non-zero submatrix in the first block-wise row is given by

$$\mathbf{A}_{11}^{(1)} = \mathbf{A}_{11}^{(2)} - \mathbf{A}_{12}^{(2)}(\mathbf{A}_{22}^{(2)})^{-1}\mathbf{A}_{21}^{(2)}. \tag{35}$$

At this point, we can solve $\mathbf{A}_{11}^{(1)}\mathbf{B}_{11} = \mathbf{I}_N$ from (32), obtaining the solution for the first block of $\mathbf{B}_1$ as

$$\mathbf{B}_{11} = (\mathbf{A}_{11}^{(1)})^{-1}. \tag{36}$$

The remaining blocks $\mathbf{B}_{k1}$ in (32) can be recursively obtained for $2 \leq k \leq K$ by:

$$\mathbf{B}_{k1} = -(\mathbf{A}_{kk}^{(k)})^{-1}\sum_{p=1}^{k-1}\mathbf{A}_{kp}^{(k)}\mathbf{B}_{p1}, \tag{37}$$

and the full matrix $\mathbf{B}_1$ is finally computed by the first node. The calculation of $\mathbf{B}_k$, for nodes $k = 2, \ldots, K$ is straightforward just using its corresponding matrix $\mathbf{I'}_k$ as defined in (31). Alternatively, every node k can use the same steps shown above just previously performing a row-wise ordering of matrix $\mathbf{A}(n)$ such that blocks $\mathbf{A}_{1i}$ and $\mathbf{A}_{ki}$ are exchanged.

B. S2: Strategy Based on LU Factorization

This strategy is based on the calculation of the matrix $\mathbf{B}_k$ by using LU factorization [41], [45]. Alternatively to the GE method, the matrices $\mathbf{B}_k$ can be obtained by computing the LU factorization of matrix $\mathbf{A}$ in (23) such that $\mathbf{A} = \mathbf{LU}$ where $\mathbf{L}$ and $\mathbf{U}$ are lower and upper triangular $KN \times KN$ matrices, respectively. This computation involves $(2/3)(KN)^3$ multiplications per iteration or $(1/3)(KN)^3$ in case we consider the symmetry property of $\mathbf{A}$ and the Cholesky decomposition is applied [41]. From (31) we derive

$$\mathbf{AB}_k = \mathbf{LUB}_k = \mathbf{I'}_k, \tag{38}$$

that can be rewritten as

$$\mathbf{UB}_k = \mathbf{L}^{-1}\mathbf{I'}_k = \mathbf{M}_k, \tag{39}$$

where $\mathbf{M}_k$ is a $KN \times N$ matrix comprised of block-wise column k of matrix $\mathbf{L}^{-1}$. By considering the block partition of $\mathbf{L}$ using submatrices of dimensions $N \times N$, we can derive the following relationship at the kth node

$$\begin{bmatrix} \mathbf{L}_{11} & \mathbf{0}_{N\times N} & \cdots & \mathbf{0}_{N\times N} \\ \mathbf{L}_{21} & \mathbf{L}_{22} & \cdots & \mathbf{0}_{N\times N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{L}_{K1} & \mathbf{L}_{K2} & \cdots & \mathbf{L}_{KK} \end{bmatrix} \begin{bmatrix} \mathbf{M}_{1k} \\ \mathbf{M}_{2k} \\ \vdots \\ \mathbf{M}_{Kk} \end{bmatrix} = \mathbf{I'}_k, \tag{40}$$

being the diagonal blocks $\mathbf{L}_{kk}$, $k = 1, \ldots, K$, lower triangular matrices. Applying forward elimination in (40), we obtain

- $\mathbf{M}_{kk} = (\mathbf{L}_{kk})^{-1}$
- $\mathbf{M}_{jk} = \mathbf{0}_{N\times N}$, for $j < k$
- $\mathbf{M}_{jk} = -(\mathbf{L}_{jj})^{-1}\sum_{p=k}^{j-1}\mathbf{L}_{jp}\mathbf{M}_{pk}$, for $j > k$.

It should be noted that the inverse matrices needed to compute $\mathbf{M}_{jk}$ can be calculated in a similar way: taking into account that the inverse of a lower triangular matrix is also lower triangular, the inverse of matrix $\mathbf{L}_{kk}$ for $k = 1, \ldots, K$, is computed by means of

$$\mathbf{L}_{kk}(\mathbf{L}_{kk})^{-1} = \begin{bmatrix} L_{kk,11} & 0 & \cdots & 0 \\ L_{kk,21} & L_{kk,22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ L_{kk,N1} & L_{kk,N2} & \cdots & L_{kk,NN} \end{bmatrix} \times \begin{bmatrix} M_{kk,11} & 0 & \cdots & 0 \\ M_{kk,21} & M_{kk,22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ M_{kk,N1} & M_{kk,N2} & \cdots & M_{kk,NN} \end{bmatrix} = \mathbf{I}_N, \tag{41}$$

resulting in

- $M_{kk,jj} = 1/L_{kk,jj}$,
- $M_{kk,ij} = -\dfrac{1}{L_{kk,ii}}\sum_{p=j}^{i-1}L_{kk,ip}M_{kk,pj}$, for $i > j$.

The values of $\mathbf{B}_k$ can be recursively calculated taking into account that $\mathbf{U}$ is an upper triangular matrix. Thus, from (39) the following system is solved at the kth node

$$\begin{bmatrix} \mathbf{U}_{11} & \mathbf{U}_{12} & \cdots & \mathbf{U}_{1K} \\ \mathbf{0}_{N\times N} & \mathbf{U}_{22} & \cdots & \mathbf{U}_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{N\times N} & \mathbf{0}_{N\times N} & \cdots & \mathbf{U}_{KK} \end{bmatrix} \begin{bmatrix} \mathbf{B}_{1k} \\ \mathbf{B}_{2k} \\ \vdots \\ \mathbf{B}_{Kk} \end{bmatrix} = \begin{bmatrix} \mathbf{M}_{1k} \\ \mathbf{M}_{2k} \\ \vdots \\ \mathbf{M}_{Kk} \end{bmatrix}, \tag{42}$$

where

- $\mathbf{B}_{Kk} = \mathbf{U}_{KK}^{-1}\mathbf{M}_{Kk}$,
- $\mathbf{B}_{pk} = \mathbf{U}_{pp}^{-1}\left(\mathbf{M}_{pk} - \sum_{j=p+1}^{K}\mathbf{U}_{pj}\mathbf{B}_{jk}\right)$, for $p < K$.

Notice that the computation of matrices $\mathbf{B}_k$ through this strategy involves the inversion of lower and upper triangular matrices of dimensions $N \times N$. We will analyze in Section V the total number of operations per node involved, but at a first glance we can state that will depend on $O(N^3)$. In this sense, this method will be efficient for small projection orders of the AP algorithm.

C. S3: Strategy Based on Matrix Inversion Lemma

Although in the two previous methods each node has to calculate only its corresponding block-wise column of matrix $\mathbf{B}$, this last strategy consider the calculation of the entire matrix $\mathbf{B}$ at each node. The way that matrix $\mathbf{A}$ is formed at each iteration allows for the use of the matrix inversion lemma, reducing the computation cost of inverting $\mathbf{A}$ and, consequently, the cost of obtaining $\mathbf{B}$. In the following we will use the discrete-time

TABLE II
SUMMARY OF THE NUMBER OF MULTIPLICATIONS TO OBTAIN $\mathbf{B}$ OR $\mathbf{B}_k$ DEPENDING ON THE STRATEGY. *DI* STANDS FOR DIRECT INVERSION OF MATRIX $\mathbf{B}$. A TYPICAL CASE WITH $N = 5$ AND $K = 4$ IS CONSIDERED FOR COMPARISON

Method	Multiplications	$N = 5, K = 4$
DI ($\mathbf{B}$)	$O((KN)^3)$	C
S1 ($\mathbf{B}_k$)	$(2K-1)O(N^3) + (2/3)N^3K(K^2-1)$	$0.42C$
S2 ($\mathbf{B}_k$)	$(1/3)\left((KN)^3 + KN(N+1)(N+2)\right) + N^3(K^2+K-1)$	$0.33C$
S3 ($\mathbf{B}$)	$O((2K)^3) + O(2K^3N^2) + O(4K^3N)$	$0.62C$

index n since this third strategy is recursive. Following $\mathbf{A}(n) = \mathbf{V}^T(n)\mathbf{V}(n)$, we can express $\mathbf{A}(n)$ from $\mathbf{A}(n-1)$ as

$$\begin{aligned}\mathbf{A}(n) &= \mathbf{A}(n-1) \\ &\quad + \mathbf{V}'(n)[\mathbf{V}'(n)]^T - \mathbf{V}'(n-L)[\mathbf{V}'(n-L)]^T \\ &= \mathbf{A}(n-1) \\ &\quad + \begin{bmatrix}\mathbf{V}'(n) & \mathbf{V}'(n-L)\end{bmatrix} \mathbf{D} \begin{bmatrix}\mathbf{V}'(n) & \mathbf{V}'(n-L)\end{bmatrix}^T \\ &= \mathbf{A}(n-1) + \mathbf{F}(n)\mathbf{D}\mathbf{F}^T(n), \end{aligned} \tag{43}$$

where

$$\mathbf{F}(n) = \begin{bmatrix}\mathbf{V}'(n) & \mathbf{V}'(n-L)\end{bmatrix}, \tag{44}$$

and $\mathbf{V}'(n)$ is a matrix of dimensions $KN \times K$ defined as

$$\mathbf{V}'(n) = \begin{bmatrix} \mathbf{v}'_{11}(n) & \mathbf{v}'_{21}(n) & \cdots & \mathbf{v}'_{K1}(n) \\ \mathbf{v}'_{12}(n) & \mathbf{v}'_{22}(n) & \cdots & \mathbf{v}'_{K2}(n) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}'_{1K}(n) & \mathbf{v}'_{2K}(n) & \cdots & \mathbf{v}'_{KK}(n) \end{bmatrix}, \tag{45}$$

where $\mathbf{v}'_{jk}(n) = \left[v_{jk}(n), v_{jk}(n-1), \ldots, v_{jk}(n-N+1)\right]^T$, that is, $\mathbf{v}'_{jk}(n)$ is the same vector as $\mathbf{v}_{jk}(n)$ in (5) but of length N instead of L, and whose elements can be computed at each node as said before. $\mathbf{D}$ is a $2K \times 2K$ matrix defined as:

$$\mathbf{D} = \begin{bmatrix} \mathbf{I}_K & \mathbf{0}_{K\times K} \\ \mathbf{0}_{K\times K} & -\mathbf{I}_K \end{bmatrix}. \tag{46}$$

Therefore, we can use the matrix inversion lemma on expression (43) and compute $\mathbf{B}(n)$ from $\mathbf{B}(n-1)$ as follows:

$$\begin{aligned}\mathbf{B}(n) &= \mathbf{B}(n-1) \\ &\quad - \mathbf{B}(n-1)\mathbf{F}(n)[\mathbf{D} + \mathbf{F}(n)^T\mathbf{B}(n-1)\mathbf{F}(n)]^{-1} \\ &\quad \times \mathbf{F}(n)^T\mathbf{B}(n-1), \end{aligned} \tag{47}$$

where the dimensions of the matrix to invert are $2K \times 2K$ and do not depend on the projection order, N. Notice that the whole matrix $\mathbf{B}$ is computed at each node in a similar way that could be computed once by the centralized MFxAP algorithm, thus, this cannot be considered a specific computation of $\mathbf{B}_k$ at each node. However, we propose this method because it is an efficient solution when the number of nodes K is small, as we will show in Section V.

V. COMPUTATIONAL COMPLEXITY

In this section we will evaluate the computational complexity of the three proposed strategies in terms of the number of multiplications per time iteration n at each node. Since the only difference among them is the calculation of the specific block $\mathbf{B}_k$ or the whole matrix $\mathbf{B}$, depending on the strategy, we will consider only the number of multiplications to obtain $\mathbf{B}_k$ or $\mathbf{B}$. The details on how the number of multiplications have been obtained for each method are given in the Appendix.

The computational cost of the three strategies in terms of multiplications per iteration is summarized in Table II, together with the cost to obtain matrix $\mathbf{B}$ by direct inversion (*DI*) of $\mathbf{A}$, which needs $O((KN)^3)$ multiplications. When $O(n^3)$ multiplications are involved, a practical number of $2n^3$ is considered. For ease of comparison, a typical case has been added in the third column with projection order $N = 5$ and $K = 4$ single-channel nodes.

Since the complexity of the three strategies depends on both N and K, Fig. 2 and 3 illustrate their impact on the computational burden of the distributed controller. Fig. 2 shows the number of multiplications required for a projection order ranging from $N = 1$ to $N = 10$ over an acoustic network formed by (a) $K = 2$ nodes and (b) $K = 8$ nodes. For projection orders higher than $N = 3$, a computational saving with respect to the direct inversion is achieved by all the strategies. Moreover, for large values of N and a small ASN (Fig. 2(a)), the computation cost required by *S3* is significantly lower than for strategies *S1* and *S2*. However, for medium size ASNs (Fig. 2(b)), the three strategies show a comparable number of operations, but significantly lower than the direct inversion.

Fig. 3 illustrates the number of multiplications required for different sizes of the acoustic network, ranging from $K = 1$ to $K = 10$ nodes for projection orders (a) $N = 2$ and (b) $N = 5$. For a small projection order (Fig. 3(a)), it can be noticed that strategy *S3* has a computational requirement higher than that of the direct inversion method. On the contrary, for high projection orders, Fig. 3(b) shows that all the proposed approaches provide a significant computational saving with respect to the direct inversion method.

In summary, for $N \geq 3$, the three proposed strategies have lower computational requirements than the *DI* method, whereas for $N = 2$, strategy *S3* is not a good option. On the other hand, strategy *S3* exhibits the best performance for high values of N. Strategies *S1* and *S2* always achieve a better performance than the *DI* method, but the cost of *S2* is always smaller than that of *S1* for all the combinations of K and N considered.

VI. SIMULATION RESULTS

In this section we present the numerical simulations carried out to evaluate the performance of the proposed EFxAP algorithm compared to two previous approximated versions of the

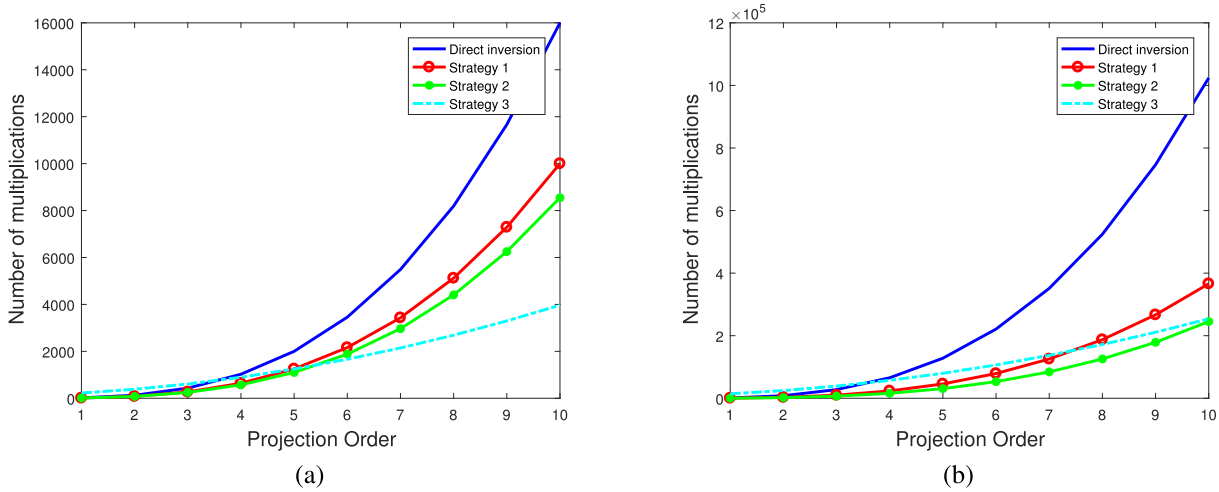

Fig. 2. Computational complexity in terms of multiplications per iteration at each node for the three strategies and the direct inversion versus the projection order N over (a) a two-node ASN and (b) an eight-node ASN.

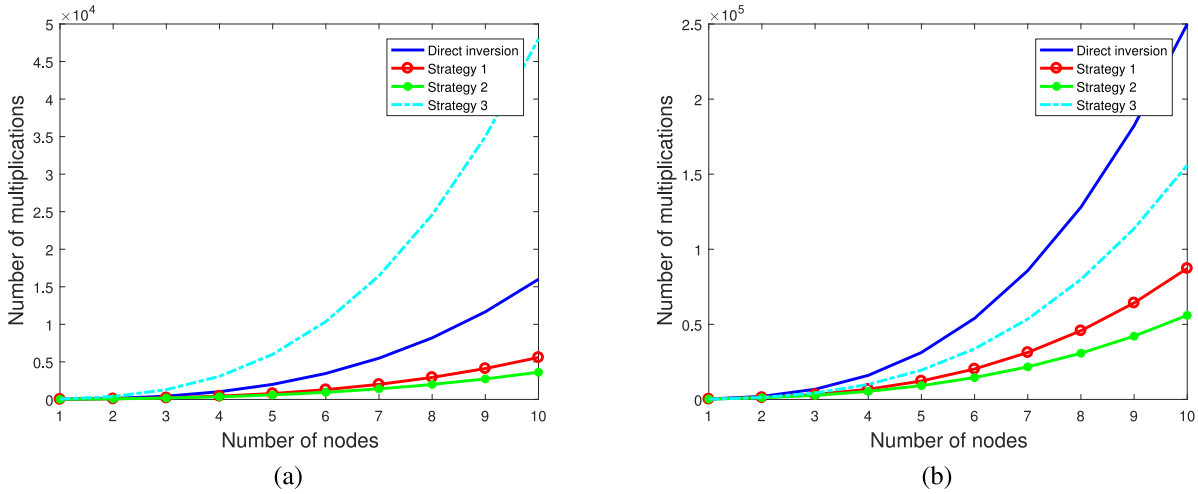

Fig. 3. Computational complexity in terms of multiplications per iteration at each node for the three strategies and the direct inversion versus the number of nodes, K, for projection orders (a) $N = 2$ and (b) $N = 5$.

distributed AP algorithm, the DFxAP [16] and the DFxAPL [15], and to the distributed multiple error FxLMS (DMEFxLMS) [14]. Since the three strategies, *S1*, *S2* and *S3*, achieve the same mathematical solution, the fastest strategy *S2* has been implemented in the simulations. Three different ASNs have been simulated comprising one node ($K = 1$), two nodes ($K = 2$), and four nodes ($K = 4$). Their setting of loudspeakers and sensors is similar to that described in [14], [16], and a sketch of the transducers is represented in Fig. 4. All the simulated ASNs use real acoustic responses measured inside the listening room of the Audio Processing Laboratory of the Institute of Telecommunications and Multimedia Applications (iTEAM). These responses have been modeled as FIR filters of $M = 256$ coefficients with a sampling rate of 2 kHz.

The ASN is deployed following a ring topology and makes use of incremental communication as shown in Fig. 1. The reference signal (unwanted noise) of the ANC system is a Gaussian noise of zero mean and unit variance that is generated by a loudspeaker located 2 meters away from both the secondary loudspeakers and the error microphones. The adaptive filters at each node have a length of $L = 150$ coefficients. The step size for each configuration and adaptive algorithm has been set by trial and error as the value that gives the fastest convergence.

A. Performance of the EFxAP Algorithm

We assume in this subsection that the underlying communication network does not suffer from any data rate or latency constraint. We also assume that the processing units of each node are powerful enough to run all the distributed algorithms in real time. The performance of the EFxAP and the other adaptive algorithms used for comparison is evaluated in terms of the instantaneous relative residual sound level at each node, $\mathrm{SL}_k(n)$, defined as the ratio in dB between the instantaneous estimated

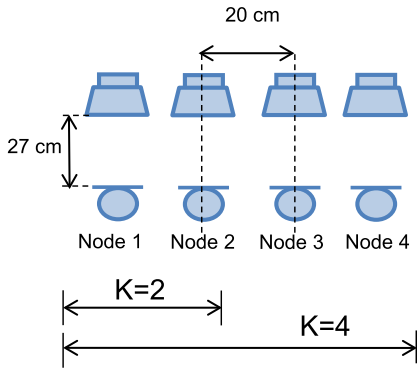

Fig. 4. Sketch of the nodes used in the experiments. Nodes selected for each ASN are indicated.

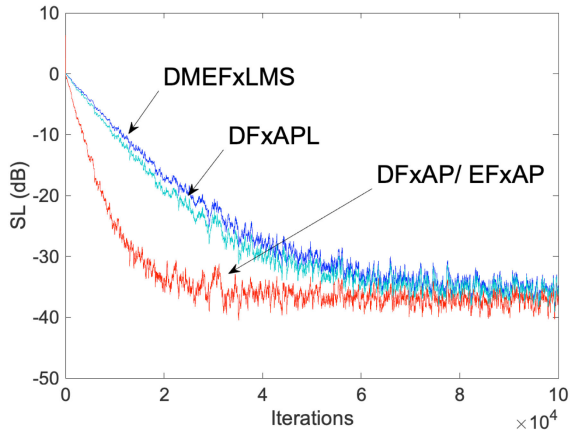

Fig. 5. Relative residual sound level obtained using a one-node ASN for the four algorithms considered with $N = 2$.

error power with and without the application of the active noise controller [16],

$$\mathrm{SL}_k(n) = 10\log_{10}\left[\frac{e_k^2(n)}{d_k^2(n)}\right], \tag{48}$$

where $d_k(n)$ is the signal measured by the microphone of the kth node when the ANC system is off, and $e_k(n)$ is the signal measured at the same microphone when the ANC system is on. In all the figures, the $\mathrm{SL}_k(n)$ is depicted versus the number of iterations providing the learning curves for each node, which have been averaged over 30 independent runs for each algorithm.

The first set of simulations is carried out considering only one node, $K = 1$, that is, one microphone and one loudspeaker, and can be seen as a baseline system for the other two network configurations. Fig. 5 shows the SL obtained by the four algorithms for this system and a small projection order, $N = 2$. As it might be expected, the EFxAP and the DFxAP exhibit equal performance since both algorithms are equivalent in a single node system. Furthermore, they exhibit a convergence behavior comparatively faster than both the DMEFxLMS and the DFxAPL, and a slightly lower final residual noise as well.

A second simulation using the two-node ASN of Fig. 4 has been carried out. Fig. 6 plots the average of $\mathrm{SL}_1(n)$ for projection orders (a) $N = 2$ and (b) $N = 5$. The results obtained by the four algorithms in the two nodes of the network are very similar, thus only the results corresponding to the first node are shown. It can be appreciated that the EFxAP algorithm outperforms the other three algorithms in the transient state for both projection orders, although its improvement in the transient state for order $N = 5$ is even more relevant (Fig. 6(b)). Regarding the noise reduction achieved at the steady state by the four algorithms, the DFxAP shows a similar value to the EFxAP, whereas the DFxAPL and DMEFxLMS algorithms exhibit a much lower performance.

Finally, the SL curves for the first and fourth node of a four-node ASN are depicted in Fig. 7 for $N = 2$ and Fig. 8 for $N = 5$. The behavior is very similar to that observed for the two-node ASN in the sense that the EFxAP outperforms the rest of the algorithms for all the nodes in both transient and steady state states. It is worth mentioning that the DMEFxLMS performance worsens compared to the two-node ASN curve shown in Fig. 6. On the other hand, the learning curves of the DFxAPL fall close to the EFxAP at the first node of the network (Fig. 7(a) and Fig. 8(a)). However, the opposite occurs at the fourth node where it is the other approximated method (DFxAP) which exhibits a performance close to the EFxAP. Therefore, we can conclude that both DFxAP and DFxAPL are not as robust as the proposed EFxAP, since their performance not only depend on the network size and the projection order, but also on the particular acoustic scenario.

B. *Discussion on the Conditions for Real Time Processing*

As said before, all the previous results have been obtained in the case of an "ideal" scenario where the data rate and latency of the underlying network fulfill the conditions for the EFxAP to work in real time. In this subsection we discuss the effect of the network latency, understood as the required time to communicate between two adjacent nodes, on the algorithm's performance. For this purpose, the sequence of operations to run the EFxAP algorithm over ASNs composed by K nodes is described in Algorithm 1. The number of multiplications required for each step is indicated as a comment at the end of the respective line.

Note that $\mathbf{A}(n)$ computation in line 17 can be recursively computed as in (43), and that although strategy *S3* does not need $\mathbf{A}(n)$ to compute $\mathbf{B}(n)$ recursively, it must compute matrix $\mathbf{F}(n)$ defined in (44). The number of multiplications in line 17 are calculated from (43) taking into account that most of the data of matrix $\mathbf{V}'(n)$ in (45) are available from the previous iteration. The number of multiplications for the rest of the operations in Algorithm 1 are straightforwardly estimated from the matrix-vector or matrix-matrix products involved.

The operations carried out by each node at one iteration start in line 7 (lines 5 and 6 are only assignments) and finish in line 25. After line 25, node k transmits its locally updated global weight vector $\mathbf{w}^{(k)}(n)$ to node $k+1$. Once all the nodes have updated their local version of the global weight vector and $k = K$, the *for* loop of lines 29–32 describes the dissemination of the global weight vector $\mathbf{w}(n)$ from node K to the rest of the nodes through incremental communication.

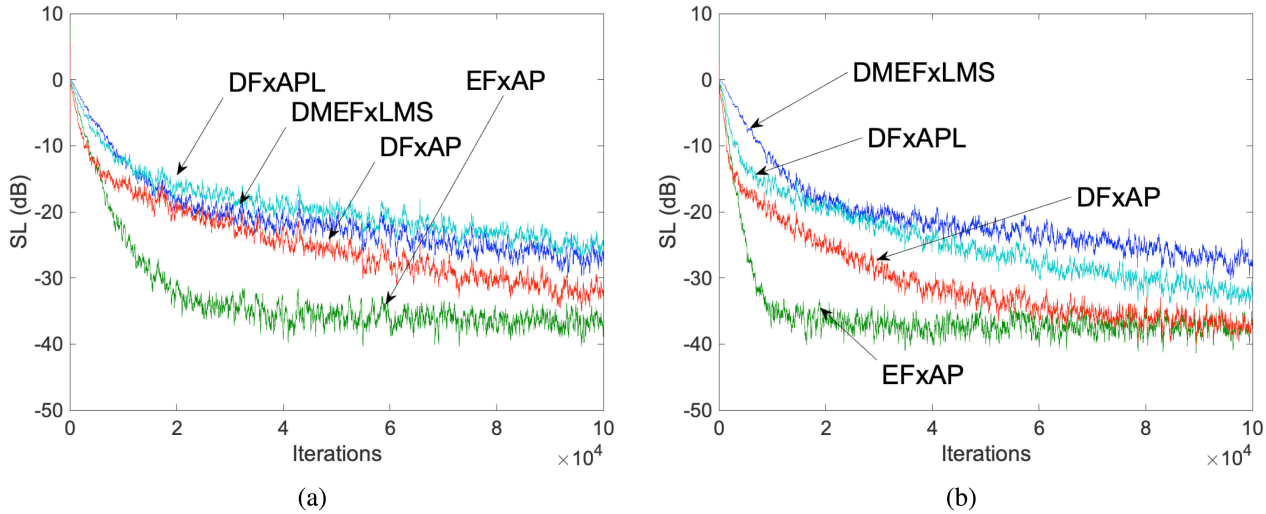

Fig. 6. Relative residual sound level obtained using a two-node ASN for the four algorithms considering only the first node with (a) $N = 2$ and (b) $N = 5$.

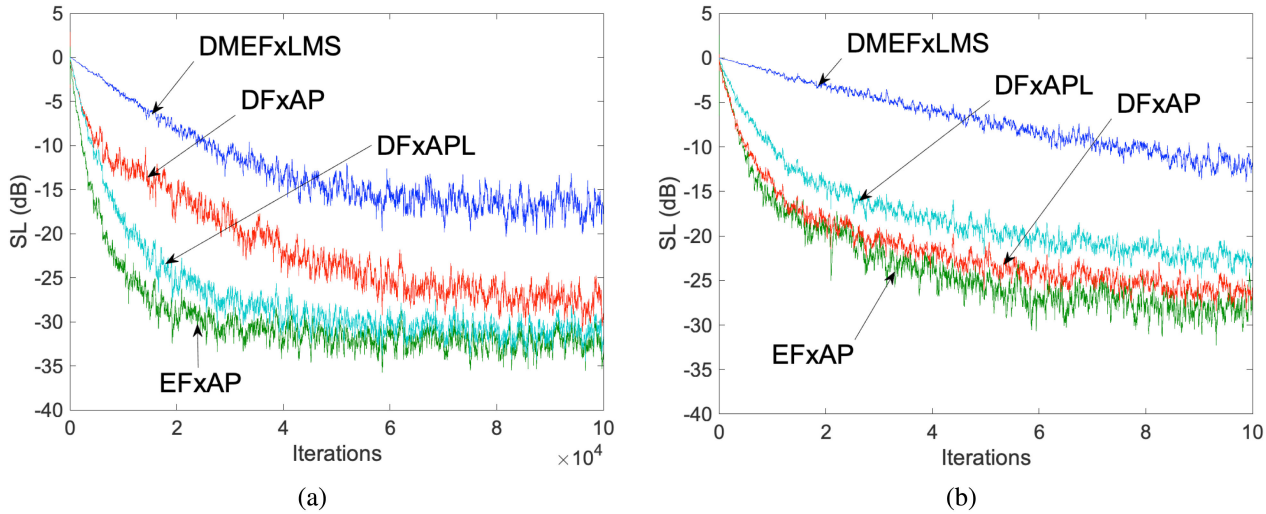

Fig. 7. Relative residual sound level obtained using a four-node ASN with $N = 2$ at the first node (a) and at the fourth node (b).

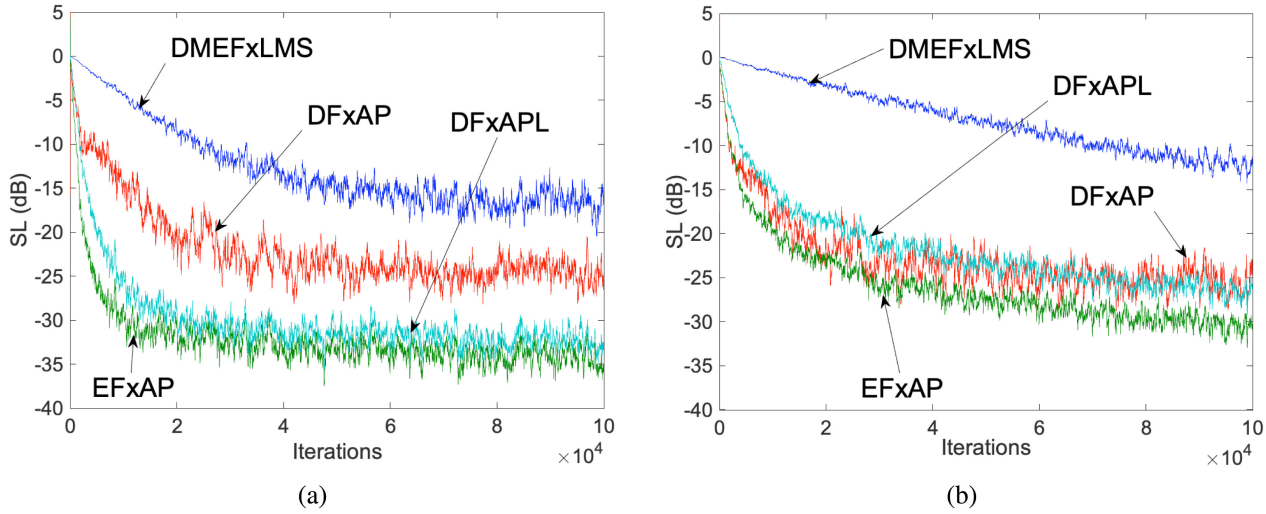

Fig. 8. Relative residual sound level obtained using a four-node ASN for the four algorithms considered with $N = 5$ at the first node (a) and at the fourth node (b).

Algorithm 1: EFxAP Algorithm.

1: Initialize: $\mathbf{w}(0) = \mathbf{w}^{(k)}(0) = [0, \ldots, 0]^T$, $\forall k$; $\mathbf{X}(0) = \mathbf{0}_{L\times M}$
2: $n = 1$ ▷ Start sample time
3: **repeat**
4: **for all** Node $1 \leq k \leq K$ **do**
5: Read sample $x(n)$
6: $\mathbf{w}_k(n) = [\mathbf{w}(n-1)]_{(L(k-1)+1:Lk)}$
7: $y_k(n) = \mathbf{w}_k^T(n)\,\mathbf{x}(n)$ ▷ L
8: **for all** $1 \leq j \leq K$ **do**
9: **for all** $1 \leq k' \leq K$ **do**
10: $\mathbf{v}_{jk'}(n) = \mathbf{X}(n)\hat{\mathbf{h}}_{jk'}$ ▷ KM
11: $\mathbf{V}_{jk'}(n) = [\mathbf{v}_{jk'}(n)\ [\mathbf{V}_{jk'}(n-1)]_{(:,1:(N-1))}]$
12: **end for**
13: **end for**
14: Obtain sample $e_k(n)$ from the kth microphone
15: $\mathbf{e}_k(n) = [e_k(n)\ e_k(n-1)\ \cdots\ e_k(n-N+1)]^T$
16: $\mathbf{V}(n) = \begin{bmatrix} \mathbf{V}_{11}(n) & \mathbf{V}_{12}(n) & \cdots & \mathbf{V}_{1K}(n) \\ \mathbf{V}_{21}(n) & \mathbf{V}_{22}(n) & \cdots & \mathbf{V}_{2K}(n) \\ \vdots & \vdots & \cdots & \vdots) \\ \mathbf{V}_{K1}(n) & \mathbf{V}_{K2}(n) & \cdots & \mathbf{V}_{KK}(n) \end{bmatrix}$
17: $\mathbf{A}(n) = [\mathbf{V}^T(n)\mathbf{V}(n) + \delta\mathbf{I}_{KN}]$ ▷ $(1+2K)(KN)^2$
18: **if** *S1* or *S2* **then**
19: Compute $\mathbf{B}_k(n)$ ▷ Multipl.: see Table II
20: **else** ▷ *S3* or *DI* cases
21: Compute $\mathbf{B}(n)$ ▷ Multipl.: see Table II
22: **end if**
23: ▷ Update global weight vector at node k
24: ▷ Assume $\mathbf{w}^{(0)}(n) = \mathbf{w}(n-1)$ for $k = 1$
25: $\mathbf{w}^{(k)}(n) = \mathbf{w}^{(k-1)}(n) - \mu\mathbf{V}(n)\mathbf{B}_k(n)\mathbf{e}_k(n)$
26: ▷ $L(KN + (KN)^2)$
27: **end for**
28: $\mathbf{w}(n) = \mathbf{w}^{(K)}(n)$ ▷ Global weight vector
29: **for all** Node $1 \leq k \leq (K-1)$ **do**
30: ▷ Disseminate $\mathbf{w}(n)$ throughout the network
31: $\mathbf{w}^{(k)}(n) = \mathbf{w}(n)$
32: **end for**
33: $n = n + 1$ ▷ Update sample time
34: **until** convergence is achieved

In order to analyze the conditions for real time processing, we have measured the execution times of the operations carried out at each node from lines 7 to 25 of Algorithm 1, without considering the communication time. Notice that the execution times will strongly depend on the hardware and software used. As an example, Table III shows the execution times T_l in μs for the same combinations of number of nodes and projection orders that have been used in Section VI-A. Sub-index l refers to the operation in the lth line of Algorithm 1. They have been measured in a personal computer running Windows 7 Enterprise (SP 1) equipped with an Intel Core i7 @ processor 2.8GHz and 8GB of RAM, and using Matlab software version 2018b. Execution times have been averaged over 100,000 iterations, discarding those 5,000 that exhibit higher values, as they were considered to be due to unwanted interruptions from the operating system.

TABLE III
EXECUTION TIMES (μS) OF THE EFXAP ALGORITHM. SUB-INDEX l IN T_l REFERS TO THE OPERATION IN THE lth LINE OF ALGORITHM 1

		K=4		K=2		K=1
		N=2	N=5	N=2	N=5	N=2
T_{proc}	T_{7-17}	4.8	7.3	4.7	4.2	1.1
	T_{19} (*S1*)	18.8	12.1	18.6	10.7	19.0
	T_{19} (*S2*)	18.8	12.0	18.6	13.7	19.0
	T_{21} (*S3*)	31.2	16.7	19.9	13.6	19.1
	T_{21} (*DI*)	51.1	73.4	23.3	45.5	19.1
T_{up}	T_{25}	4.1	4.7	4.0	4.7	6.5

The first row in Table III shows the execution time T_{7-17} to carry out operations from lines 7 to 17 as a whole. The next four rows show the execution times taken by *S1* and *S2* strategies to compute $\mathbf{B}_k(n)$, and the execution times taken by *S3* and the *DI* method to compute $\mathbf{B}(n)$, respectively. The last row of Table III shows the execution time T_{25} of the filter updating carried out in line 25 of Algorithm 1. As it can be seen, for a particular combination of number of nodes K and projection order N (comparisons along a column), the *DI* method always requires the highest computation time compared to *S1*-*S3*, except for the case of $K = 1$ (one node), which is equal to the other three methods. Additionally, *S3* usually requires a higher computation time compared to *S1* and *S2* strategies. On the other hand, notice that the times T_{19} and T_{21} taken by a particular strategy for different values of K and N (comparisons along a row), do not vary accordingly to the required number of operations calculated in Table II. As said before, the execution times extremely depend on the hardware and software used, thus the only valid comparison in Table III is between strategies for a same value of K and N.

Finally, we have grouped the executions times due to operations that a node can perform with local signals and data as T_{proc}, as shown in the first column of Table III. Note that its value will depend on K, N and the adopted strategy (lines 19 or 21). We have also denoted by T_{up} the execution time of line 25 where the filter is updated, separated from T_{proc}, because at this point current node k needs the value $\mathbf{w}^{(k-1)}(n)$ from the previous node to perform the operation. At this point, we introduce the effect of the network latency as a third execution time denoted by T_{tx}, which is the time required to communicate the data between adjacent nodes.

To better understand the sequencing of operations at each node as well as the sequencing of the whole network, Fig. 9 shows the time diagram of Algorithm 1 of an acoustic network composed of k nodes, where times T_{proc}, T_{up} and T_{tx} are depicted in blue, red and green color respectively. We assume that all the nodes in the network have similar characteristics, thus T_{proc}, T_{up} and T_{tx} are depicted of the same duration independently of the node. As shown in Fig. 9, the total execution time of the distributed EFxAP algorithm can be computed as $T_{\text{Total}} = T_{\text{proc}} + K(T_{\text{up}} + T_{\text{tx}}) + (K-1)T_{\text{tx}}$. Considering the execution times shown for example in Table III for the case of $K = 4$ and $N = 5$, $T_{\text{proc}} = 19.3$ μs if the *S2* strategy is adopted. To estimate T_{tx}, consider that the adaptive filters have $L = 150$ coefficients, so the global vector has $LK = 600$ elements that

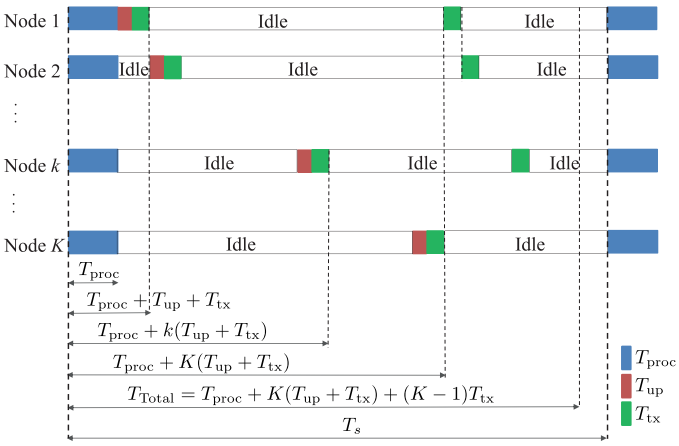

Fig. 9. Time diagram of Algorithm 1 for an ASN composed of k nodes.

have to be transmitted between consecutive nodes. If any element is coded with 16 bits and a 100Mbits/s network is used, then $T_{\text{tx}} = 600 \cdot 16/10^8 = 96\,\mu$s. Therefore, the total execution time for each iteration would be $T_{\text{Total}} = 19.3 + 4 \cdot (4.7 + 96) + 3 \cdot 96 = 710.1\ \mu$s.

If $T_{\text{Total}} \leq T_s$, being T_s the sampling time, the EFxAP algorithm will run in real time and the initial hypotheses assumed in terms of latency and data-rate constraints will be fulfilled. For the example above where $T_{\text{Total}} = 710.1\ \mu$s, this could be accomplished using a sampling rate below 1408 Hz, which is usually enough for ANC applications. Otherwise, if $T_{\text{Total}} > T_s$, the EFxAP algorithm would be forced to perform the filter updating every two, three or more sampling times. Taking the same approach as in [14] and assuming homogeneous nodes, the latency can be modeled through a constant parameter p such that pT_s is the time to transmit data between two consecutive nodes, that is, $T_{\text{tx}} = pT_s$, where p is a small positive integer ($p = 1, 2, 3, \ldots$). Therefore, a node receives the information from its precedent node every KpT_s s as it can be seen from Fig. 9, i.e., KpT_s is the time required to complete a whole round of the incremental network. It was shown in [14] that the convergence speed of the DMEFxLMS algorithm decreased with the value of p, or that even it diverged when p was large enough. With this approach in mind, we have tested the EFxAP and the rest of the distributed AP algorithms of Section VI-A assuming the global weight vector $\mathbf{w}(n)$ is updated every KpT_s s instead of every T_s s, and we have obtained similar results to those shown in [14]: all the algorithms degraded in a similar way showing a slower convergence, or even diverging, as p increases. However, it can be stated that strategies *S1*-*S3* proposed in this work will obtain better performance than the direct inversion of $\mathbf{A}(n)$ for the same experienced latency since their T_{proc} values will always be shorter.

Summarizing the influence of the network latency on the distributed EFxAP algorithm, for a given combination of K, N and adopted strategy, if the communication time between nodes is such that the total execution time $T_{\text{Total}} \leq T_s$, the EFxAP algorithm will run on real time and it will achieve the performance results shown in Section VI-A. Otherwise, its convergence will become slower, or even will be put at risk, but the same effect would experience any distributed AP-type and LMS-type adaptive algorithm running over the same ASN.

VII. Conclusion

In this paper, an exact distributed version of the centralized multichannel filtered-x AP algorithm for ANC over acoustic networks has been presented. Denoted by EFxAP, this algorithm achieves the same solution as the centralized algorithm over ASNs with a ring topology and incremental communication, providing that there are no latency or data rate constraints in the underlying network. We have also presented three different strategies to reduce the computational burden of the EFxAP at each node: the Gauss elimination (*S1*), the block LU factorization (*S2*) and the matrix inversion lemma (*S3*). The first two strategies, *S1* and *S2*, distribute the computational burden due to a matrix inversion of size $KN \times KN$ among the network nodes, whereas the *S3* strategy reduces the number of operations required at each node to compute the full inverse matrix. We have detailed the computational cost of every strategy and we have concluded that for projections orders such that $N \geq 3$, all the strategies require a significant lower number of operations compared to the direct matrix inversion. In order to evaluate the performance of the EFxAP for ANC applications, we have carried out numerical experiments comparing the sound level reduction obtained by the EFxAP to that achieved by two previous approximated distributed AP algorithms and by the distributed LMS solution. It has been shown that the proposed EFxAP exhibits the best performance in transient and steady states for ASNs with 1, 2 and 4 nodes, and for low and high projection orders as well. Finally, we have discussed the effect of the network latency, understood as the required communication time between consecutive nodes, on the EFxAP algorithm concluding that, for the same latency constraints, the proposed strategies will perform better than the direct inversion of a full matrix.

Appendix

The number of operations required by each strategy is derived in the following.

A. Computations Required by S1

In the case of the EFxAP solution obtained by Gaussian elimination, the number of multiplications is derived from the method developed in Section IV-A:

- Block matrix reduction: each lower block needs $2N^3$ multiplications and one matrix inversion of size $N \times N$, which means $2N^3(k-1)^2 + O(N^3)$ multiplications. However, since the same matrix multiplication is repeated at each column or row, its cost is halved resulting in $N^3(k-1)^2 + O(N^3)$ multiplications.
- Considering all the reductions from K to 2 yields:

$$(K-1)O(N^3) + 2N^3 \sum_{k=2}^{K} (k-1)^2$$

- The computation of the matrices $\mathbf{B}_k$ requires the following number of operations: $\mathbf{B}_{1k}$ requires $O(N^3)$ multiplications, $\mathbf{B}_{2k}$ requires $O(N^3) + 2N^3$ and so on. Consequently $\mathbf{B}_{pk}$ requires $O(N^3) + (p-1)2N^3$. Thus, for the K matrices $\mathbf{B}_{pk}$ we need $KO(N^3) + 2N^3 \sum_{k=2}^{K}(k-1)$ multiplications.

Summing up all the above operations, we can state the following number of multiplications for strategy *S1*:

$$(2K-1)O(N^3) + 2N^3 \sum_{k=2}^{K} \left((k-1)^2 + (k-1)\right).$$

The sum term is equivalent to $\sum_{k=1}^{K-1}(k(k+1))$, which once summed up simplifies the above formula as

$$(2K-1)O(N^3) + \frac{2}{3}N^3K(K^2-1).$$

B. Computations Required by S2

Considering now strategy *S2*, its computational cost at each node is derived from Section IV-B:

- LU decomposition: $(1/3)(KN)^3$ using Cholesky.
- $\mathbf{M}_k$ matrix calculation: considering the multiplications needed by the node with the highest computational cost requirements (first node):

$$K\left[1 + \frac{1}{2}\sum_{m=2}^{N}(m(m+1))\right] + ((K-1)(K+2)/2)N^3$$

- $\mathbf{B}_k$ matrix calculation requires

$$K\left[1 + \frac{1}{2}\sum_{m=2}^{N}(m(m+1))\right] + (K(K+1)/2)N^3$$

multiplications.

Therefore, the total number of multiplications required at each node for strategy *S2* is given by

$$\frac{1}{3}(KN)^3 + 2K\left[1 + \frac{1}{2}\sum_{m=2}^{N}(m(m+1))\right] + \frac{1}{2}N^3((K-1)(K+2) + K(K+1)),$$

and after executing the sum term, it can be expressed as

$$\frac{1}{3}(KN)^3 + \frac{1}{3}KN(N+1)(N+2) + N^3(K^2+K-1).$$

C. Computations Required by S3

In the case of the EFxAP solution based on strategy in Section IV-C, each node calculates the whole inverse matrix $\mathbf{B}(n)$ using the matrix inversion lemma. This method seems convenient for small networks with only a few nodes and an AP algorithm with high projection order as it computes the inverse of a $2K \times 2K$ matrix instead of a matrix of size $KN \times KN$. On the other hand, a precise computation of the first value of $\mathbf{B}(n)$ is required since it is a recursive method.

Derived from equations in Section IV-C, the cost of this method is given by $O((2K)^3) + O(2K^3N^2) + O(4K^3N)$ where the second term corresponds to the matrix product of $\mathbf{F}^T(n)\mathbf{B}(n-1)$ and the third term to the product of the resulting $2K \times KN$ matrix with $\mathbf{F}(n)$.

REFERENCES

[1] A. Bertrand, "Applications and trends in wireless acoustic sensor networks: A signal processing perspective," in *Proc. 18th IEEE Symp. Commun. Veh. Technol. Benelux (SCVT)*, 2011, pp. 1–6.

[2] Á. Ledeczi, T. Hay, P. Volgyesi, D. R. Hay, A. Nadas, and S. Jayaraman, "Wireless acoustic emission sensor network for structural monitoring," *IEEE Sensors J.*, vol. 9, no. 11, pp. 1370–1377, Nov. 2009.

[3] P. Heilmann, R. Weiss, R. Weigel, and L. Schwarz, "Emission monitoring of machines using equally distributed wireless acoustic sensor nodes," in *Proc. IEEE SENSORS*, 2017, pp. 1–3.

[4] F. Alías and R. M. Alsina-Pagès, "Review of wireless acoustic sensor networks for environmental noise monitoring in smart cities," *J. Sensors*, vol. 2019, no. ID 7634860, 2019, Art. no. 13.

[5] M. Cobos, J. J. Perez-Solano, S. Felici-Castell, J. Segura, and J. M. Navarro, "Cumulative-sum-based localization of sound events in low-cost wireless acoustic sensor networks," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 12, pp. 1792–1802, Dec. 2014.

[6] A. Brendel and W. Kellermann, "Distributed source localization in acoustic sensor networks using the coherent-to-diffuse power ratio," *IEEE J. Sel. Top. Signal Process.*, vol. 13, no. 1, pp. 61–75, Feb. 2019.

[7] S. Al-Sayed, J. Plata-Chaves, M. Muma, M. Moonen, and A. M. Zoubir, "Node-specific diffusion LMS-based distributed detection over adaptive networks," *IEEE Trans. Signal Process.*, vol. 66, no. 3, pp. 682–697, Feb. 2018.

[8] L. Wang, T.-K. Hon, J. D. Reiss, and A. Cavallaro, "Self-localization of ad-hoc arrays using time difference of arrivals," *IEEE Trans. Signal Process.*, vol. 64, no. 4, pp. 1018–1033, Feb. 2016.

[9] A. Alexandridis and A. Mouchtaris, "Multiple sound source location estimation in wireless acoustic sensor networks using DOA estimates: The data-association problem," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 2, pp. 342–356, Feb. 2018.

[10] A. Brendel and W. Kellermann, "Localization of multiple simultaneously active sources in acoustic sensor networks using ADP," in *Proc. IEEE 7th Int. Workshop Comput. Adv. Multi-Sensor Adaptive Process.*, 2017, pp. 1–5.

[11] A. I. Koutrouvelis, T. W. Sherson, R. Heusdens, and R. C. Hendriks, "A low-cost robust distributed linearly constrained beamformer for wireless acoustic sensor networks with arbitrary topology," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 26, no. 8, pp. 1434–1448, Aug. 2018.

[12] X. Leng, J. Chen, J. Benesty, and I. Cohen, "On speech enhancement using microphone arrays in the presence of co-directional interference," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2018, pp. 511–515.

[13] D. Cherkassky and S. Gannot, "Blind synchronization in wireless acoustic sensor networks," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 25, no. 3, pp. 651–661, Mar. 2017.

[14] M. Ferrer, M. de Diego, G. Piñero, and A. Gonzalez, "Active noise control over adaptive distributed networks," *Signal Process.*, vol. 107, pp. 82–95, 2015.

[15] C. Antoñanzas, M. Ferrer, M. de Diego, and A. Gonzalez, "Affine-projection-like algorithm for active noise control over distributed networks," in *IEEE 9th Sensor Array Multichannel Signal Process. Workshop (SAM)*, 2016, pp. 1–5.

[16] M. Ferrer, A. Gonzalez, M. de Diego, and G. Piñero, "Distributed affine projection algorithm over acoustically coupled sensor networks," *IEEE Trans. Signal Process.*, vol. 65, no. 24, pp. 6423–6434, Dec. 2017.

[17] Y. Chu, C. Mak, Y. Zhao, S. Chan, and M. Wu, "Performance analysis of a diffusion control method for anc systems and the network design," *J. Sound Vib.*, vol. 475, 2020, Art. no. 115273.

[18] S. Elliott, I. Stothers, and P. Nelson, "A multiple error LMS algorithm and its application to the active control of sound and vibration," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 35, no. 10, pp. 1423–1434, Oct. 1987.

[19] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Proc. Electron. Commun. Jpn.*, vol. J67-A, pp. 126–132, 1984.

[20] G. Rombouts and M. Moonen, "A sparse block exact affine projection algorithm," *IEEE Trans. Acoust., Speech Signal Process.*, vol. 10, no. 2, pp. 100–108, Feb. 2002.

[21] Y. Choi, H. Shin, and W. Song, "Adaptive regularization matrix for affine projection algorithm," *IEEE Trans. Circuits Syst.*, vol. 54, no. 12, pp. 1087–1091, Dec. 2007.
[22] C. Paleologu, J. Benesty, and S. Ciochina, "A variable step-size affine projection algorithm designed for acoustic echo cancellation," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 8, pp. 1466–1478, Nov. 2008.
[23] J. Lee, Y. Park, and D. H. Youn, "Robust pseudo affine projection algorithm with variable step-size," *Electron. Lett.*, vol. 44, no. 3, pp. 250–251, 2008.
[24] L. Liu *et al.*, "A variable step-size proportionate affine projection algorithm for identification of sparse impulse response," *EURASIP J. Adv. Signal Process.*, 2009, Art. no. 150914.
[25] S. Kim, S. Kong, and W. Song, "An affine projection algorithm with evolving order," *IEEE Signal Process. Lett.*, vol. 16, no. 11, pp. 937–940, Jul. 2009.
[26] K. Mayyas, "A variable step-size affine projection algorithm," *Digit. Signal Process.*, vol. 20, no. 2, pp. 502–510, 2010.
[27] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*, Englewood Cliffs, NJ, USA: Prentice-Hall, 1985.
[28] S. Douglas, "The fast affine projection algorithm for active noise control," in *Proc. 29th Asilomar Conf. Signals Syst. Compon.*, 1995, pp. 1245–1249.
[29] M. Bouchard, "Multichannel affine and fast affine projection algorithms for active noise control and acoustic equalization systems," *IEEE Trans. Speech Audio Process.*, vol. 11, no. 1, pp. 54–60, Jan. 2003.
[30] M. Ferrer, M. de Diego, A. Gonzalez, and G. Piñero, "Efficient implementation of the affine projection algorithms for active noise control aplication," in *Proc. 12th Eur. Signal Process. Conf. (EUSIPCO)*, 2004, pp. 929–932.
[31] A. Carini and G. Sicuranza, "Transient and steady-state analysis of filtered-x affine projection algorithms," *IEEE Trans. Signal Process.*, vol. 54, no. 2, pp. 665–678, Feb. 2006.
[32] M. Ferrer, A. Gonzalez, M. de Diego, and G. Piñero, "Fast affine projection algorithms for filtered-x multichannel active noise control," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 16, no. 8, pp. 1396–1408, Nov. 2008.
[33] M. Ferrer, A. Gonzalez, M. de Diego, and G. Piñero, "Transient analysis of the conventional filtered-x affine projection algorithm for active noise control," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 3, pp. 652–657, Mar. 2011.
[34] M. Ferrer, M. de Diego, A. Gonzalez, and G. Piñero, "Steady-state mean square performance of the multichannel filtered-x affine projection algorithm," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2771–2785, Jun. 2012.
[35] S. J. Elliott and C. C. Boucher, "Interaction between multiple feedforward active control systems," *IEEE Trans. Speech Audio Process.*, vol. 2, no. 4, pp. 521–530, Oct. 1994.
[36] N. George and G. Panda, "A particle-swarm-optimization-based decentralized nonlinear active noise control system," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 12, pp. 3378–3386, Jul. 2012.
[37] J. A. Mosquera-Sánchez, W. Desmet, and L. P. de Oliveira, "Multichannel feedforward control schemes with coupling compensation for active sound profiling," *J. Sound Vib.*, vol. 396, pp. 1–29, 2017.
[38] G. Zhang, J. Tao, X. Qiu, and I. Burnett, "Decentralized two-channel active noise control for single frequency by shaping matrix eigenvalues," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 1, pp. 44–52, Jan. 2019.
[39] C. Lopes and A. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Trans. Signal Process.*, vol. 55, no. 8, pp. 4064–4077, Aug. 2007.
[40] A. Bertrand, S. Doclo, S. Gannot, N. Ono, and T. van Waterschoot, "Special issue on wireless acoustic sensor networks and ad hoc microphone arrays," *Signal Process.*, vol. 107, pp. 1–3, 2015.
[41] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 4th ed., Baltimore, MD, USA: John Hopkins Univ. Press, 2013.
[42] S. Haykin, *Adaptive Filter Theory*, 4th ed., Englewood Cliffs, NJ, USA: Prentice-Hall, 2002.
[43] J. Plata-Chaves, A. Bertrand, and M. Moonen, "Incremental multiple error filtered-x LMS for node-specific active noise control over wireless acoustic sensor networks," in *Proc. IEEE Sensor Array Multichannel Signal Process. Workshop (SAM)*. 2016, pp. 1–5.
[44] C. Paleologu, J. Benesty, and S. Ciochina, "Regularization of the affine projection algorithm," *IEEE Trans. Circuits Syst. II: Express Briefs*, vol. 58, no. 6, pp. 366–370, Jun. 2011.
[45] N. J. Higham, "Gaussian elimination," *Wiley Interdiscipl. Rev.: Comput. Statist.*, vol. 3, no. 3, pp. 230–238, 2011.