DISTRIBUTED ADAPTIVE EIGENVECTOR ESTIMATION OF THE SENSOR SIGNAL COVARIANCE MATRIX IN A FULLY CONNECTED SENSOR NETWORK

Alexander Bertrand[*]*, Marc Moonen*

KU Leuven - Dept. ESAT/SCD & iMinds Future Health Department
Kasteelpark Arenberg 10, B-3001 Leuven, Belgium
E-mail: alexander.bertrand@esat.kuleuven.be; marc.moonen@esat.kuleuven.be

ABSTRACT

In this paper, we describe a distributed adaptive (time-recursive) algorithm to estimate and track the eigenvectors corresponding to the Q largest or smallest eigenvalues of the global sensor signal covariance matrix in a wireless sensor network (WSN). We only address the case of fully connected (broadcast) networks, in which the nodes broadcast compressed Q-dimensional sensor observations. It can be shown that the algorithm converges to the desired eigenvectors without explicitely constructing the global covariance matrix that actually defines them, i.e., without the need to centralize all the raw sensor observations. The algorithm allows each node to estimate (a) the node-specific entries of the global covariance matrix eigenvectors, and (b) Q-dimensional observations of the full set of sensor observations projected onto the Q estimated eigenvectors. The theoretical results are validated by means of numerical simulations.

Index Terms— Wireless sensor networks, distributed estimation, distributed compression, distributed eigenvector estimation.

1. INTRODUCTION

The eigenvectors of a covariance matrix play a crucial role in several algorithms and applications, including principal component analysis (PCA), the Karhunen Loeve transform (KLT), steering vector estimation, total least squares estimation and nullspace/subspace estimation. In this paper, we consider a set of spatially distributed wirelessly connected sensor nodes where a node k collects observations of a node-specific stochastic vector $\mathbf{y}_k$. Let $\mathbf{y}$ be the global vector in which all $\mathbf{y}_k$'s are stacked, then we aim to estimate and track Q eigenvectors of the global covariance matrix corresponding to $\mathbf{y}$. In principle, this would require each node to transmit its raw sensor observations to a central node or fusion center (FC), where the global covariance matrix can be constructed, after which an eigenvalue decomposition (EVD) can be performed. However, transmitting raw observations of the different $\mathbf{y}_k$'s to an FC may require too much communication bandwidth (in particular if observations are collected at a high sampling rate, as in audio or video applications). Furthermore, if $\mathbf{y}$ has a large dimension, the global covariance matrix may become too large to process in the FC.

To avoid these issues, we propose a distributed algorithm to estimate and track the Q eigenvectors without explicitely constructing the global covariance matrix that actually defines them, i.e., without the need to gather all the sensor observations in an FC. The algorithm is referred to as the distributed adaptive covariance matrix eigenvector estimation (DACMEE) algorithm. Instead of transmitting all raw sensor observations to an FC, the DACMEE algorithm lets each node broadcast Q-dimensional (compressed) observations to all the other nodes in the network. The shared data actually corresponds to the sensor observations projected onto the Q estimated eigenvectors (which can then be used, e.g., for compression/decompression based on PCA or KLT, once the DACMEE algorithm has converged). It is noted that the fully connected network case is considered here merely for the sake of an easy exposition, as a similar algorithm can also be defined in partially connected networks (details omitted).

Relation to prior work: Two different cases are mainly considered in the literature where either (a) all the nodes collect observations of the full vector $\mathbf{y}$, or (b) each node collects observations of a node-specific subset of the entries of $\mathbf{y}$ (as it is the case in this paper). Let $\mathbf{Y}$ denote an $M \times N$ observation matrix containing N observations of an M-dimensional stochastic vector $\mathbf{y}$, then (a) corresponds to the case where the *columns* of $\mathbf{Y}$ are distributed over the different nodes, whereas in case (b), the *rows* of $\mathbf{Y}$ are distributed over the nodes. The cases (a) and (b) are very different in nature and are tackled in different ways.

Case (a) is addressed in [1–3] for ad hoc topologies and in [4] for a fully connected topology. In [1], the global sample covariance matrix is first computed by means of a consensus averaging (CA) algorithm that exchanges $M \times M$ matrices in each iteration, after which each node can perform a local EVD. If only a subset of the eigenvectors[1] is desired, one can use distributed optimization techniques in which only M-dimensional vectors are exchanged between nodes [2,3]. In [4], a distributed QR decomposition is performed followed by an EVD, in a fully connected network.

Case (b) is considered to be more challenging, as it requires to capture the cross-correlation between observations in different node pairs. This case is tackled in [5, 6] (only for the case of principal eigenvectors), again by means of CA techniques. However, these algorithms require nested loops where the inner loop performs many CA iterations with a full reset for each outer loop iteration (and each new observation of $\mathbf{y}$), resulting in a relatively large communication load since each node transmits more data than actually collected by its sensors. Furthermore, this inner loop reset also hampers adaptive or time-recursive implementations. Finally, it is noted that there exists other related work in the context of (b) (see, e.g., [7, 8]), which however requires prior knowledge of the global covariance matrix,

[*]The work of A. Bertrand was supported by a Postdoctoral Fellowship of the Research Foundation - Flanders (FWO). This work was carried out at the ESAT Laboratory of KU Leuven, in the frame of KU Leuven Research Council CoE EF/05/006 'Optimization in Engineering' (OPTEC) and PFV/10/002 (OPTEC), Concerted Research Action GOA-MaNet, the Belgian Programme on Interuniversity Attraction Poles initiated by the Belgian Federal Science Policy Office IUAP P7/23, Research Project iMinds, and Research Project FWO nr. G.0763.12 'Wireless acoustic sensor networks for extended auditory communication'. The scientific responsibility is assumed by its authors.

[1] [2, 3] focuses on the eigenvector corresponding to the smallest eigenvalue of the covariance matrix, but the algorithm is easily adapted to compute the prinicipal eigenvectors.

which is assumed to be unknown here.

2. PROBLEM STATEMENT

Consider a fully connected wireless broadcast sensor network with a set of sensor nodes $\mathcal{K} = \{1, \ldots, K\}$. Node k collects observations of a complex-valued M_k-dimensional stochastic vector $\mathbf{y}_k$, which is assumed to be (short-term[2]) stationary and ergodic. We define the M-dimensional stochastic vector $\mathbf{y}$ as the stacked version of all $\mathbf{y}_k$'s, where $M = \sum_{k\in\mathcal{K}} M_k$. The covariance matrix of $\mathbf{y}$ is defined as

$$\mathbf{R}_{yy} = E\{(\mathbf{y} - E\{\mathbf{y}\})(\mathbf{y} - E\{\mathbf{y}\})^H\} \tag{1}$$

where $E\{.\}$ denotes the expected value operator and where the superscript H denotes the conjugate transpose operator. Without loss of generality, we assume that $\mathbf{y}$ is zero-mean, hence, $\mathbf{R}_{yy} = E\{\mathbf{y}\mathbf{y}^H\}$, which may require a mean subtraction preprocessing step. Ergodicity of $\mathbf{y}$ implies that $\mathbf{R}_{yy}$ can be approximated from N observations as

$$\mathbf{R}_{yy} \approx \frac{1}{N}\sum_{t=1}^{N} \mathbf{y}[t]\mathbf{y}[t]^H \tag{2}$$

where $\mathbf{y}[t]$ is an observation of $\mathbf{y}$ at sample time t.

The eigenvalue decomposition (EVD) of $\mathbf{R}_{yy}$ is defined as

$$\mathbf{R}_{yy} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^H \tag{3}$$

where $\mathbf{\Sigma} = \text{diag}(\lambda_1, \ldots, \lambda_K)$ is a real diagonal matrix with the eigenvalues as its diagonal elements (sorted in decreasing order of magnitude), and where the unitary matrix $\mathbf{U}$ contains the corresponding eigenvectors in its columns. Let $\hat{\mathbf{X}}$ denote the $M \times Q$ matrix which contains the first Q principal eigenvectors of $\mathbf{R}_{yy}$ in its columns, i.e., the eigenvectors corresponding to the Q largest eigenvalues:

$$\hat{\mathbf{X}} = \mathbf{U}\begin{bmatrix} \mathbf{I}_Q \\ \mathbf{O}_{(M-Q)\times Q} \end{bmatrix} \tag{4}$$

where $\mathbf{I}_Q$ denotes the $Q \times Q$ identity matrix and $\mathbf{O}_{(M-Q)\times Q}$ denotes the $(M-Q) \times Q$ all-zero matrix. The Q principal eigenvectors in $\hat{\mathbf{X}}$ can be used in a context of principal component analysis (PCA), to compute a rank-Q approximation of $\mathbf{R}_{yy}$, or for compression/decompression of the observations of $\mathbf{y}$ (based on the KLT). It is noted that, even though we focus here on the principal eigenvectors, all results in this paper can be straighforwardly modified to compute the last Q columns of $\mathbf{U}$ instead, i.e., the eigenvectors corresponding to the Q smallest eigenvalues. The latter can be used for, e.g., nullspace tracking or for total least squares estimation [9].

To estimate and track $\hat{\mathbf{X}}$, all nodes may transmit their observations to an FC, where $\mathbf{R}_{yy}$ can be constructed and updated at regular time intervals (e.g., based on (2) using the N most recent observations), followed by the computation of (3)-(4). However, if the dimensions of the $\mathbf{y}_k$'s are large, transmitting all the raw observations to the FC requires a large communication bandwidth, and the computation of (2) and (3)-(4) requires a significant computational power at the FC.

3. THE DACMEE ALGORITHM IN FULLY-CONNECTED NETWORKS

In this section, we present a distributed adaptive (time-recursive) algorithm where each node is responsible for estimating a specific part of $\hat{\mathbf{X}}$, and where the centralized (off-line) construction of the full covariance matrix $\mathbf{R}_{yy}$ is avoided. The algorithm is referred to as the distributed adaptive covariance matrix eigenvector estimation (DACMEE) algorithm. In the DACMEE algorithm, each node broadcasts only Q-dimensional (compressed) observations, which significantly reduces the communication bandwidth if $Q \ll M_k$.

3.1. Preliminaries

The DACMEE algorithm is an iterative algorithm that updates the $M \times Q$ matrix $\mathbf{X}^i$, where i is the iteration index (i typically runs N times slower than the sampling of the sensors, where N is chosen large enough such that (2) is sufficiently accurate, see also Subsection 3.2). We define the block partitioning[3]

$$\mathbf{X}^i = \begin{bmatrix} \mathbf{X}_1^i \\ \vdots \\ \mathbf{X}_K^i \end{bmatrix} \tag{5}$$

where the $M_k \times Q$ block matrix $\mathbf{X}_k^i$ is updated by node k. The general goal is that $\lim_{i\to\infty} \mathbf{X}^i = \hat{\mathbf{X}}$, by letting nodes exchange compressed observations of their $\mathbf{y}_k$'s. To this end, the compression matrix that is used in between iteration i and iteration $i+1$ at node k is chosen as $\mathbf{X}_k^i$, hence the latter serves both as an estimation variable and a compression matrix. The compressed versions of the $\mathbf{y}_k$'s are denoted as

$$\mathbf{z}_k^i = \mathbf{X}_k^{i\,H}\mathbf{y}_k \tag{6}$$

and we define $\mathbf{z}_{-k}^i$ as the stacked version of all the $\mathbf{z}_q^i$'s, $\forall\ q \in \mathcal{K}\backslash\{k\}$, i.e., $\mathbf{z}_{-k}^i = [\mathbf{z}_1^{i\,T}\ \ldots\ \mathbf{z}_{k-1}^{i\,T}\ \mathbf{z}_{k+1}^{i\,T}\ \ldots\ \mathbf{z}_K^{i\,T}]^T$. Since the network is fully connected, node k collects observations of the following stochastic vector

$$\widetilde{\mathbf{y}}_k^i = \begin{bmatrix} \mathbf{y}_k \\ \mathbf{z}_{-k}^i \end{bmatrix} \tag{7}$$

with corresponding covariance matrix

$$\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i = E\{\widetilde{\mathbf{y}}_k^i \widetilde{\mathbf{y}}_k^{i\,H}\}\ . \tag{8}$$

For the sake of an easy notation, we define the matrix $\mathbf{C}_k^i$ that allows to write $\widetilde{\mathbf{y}}_k^i$ as a function of the global $\mathbf{y}$, i.e.,

$$\widetilde{\mathbf{y}}_k^i = \mathbf{C}_k^{i\,H}\mathbf{y} \tag{9}$$

with

$$\mathbf{C}_k^i = \left[\begin{array}{c|c} \mathbf{O}_{\underline{S}_k\times M_k} & \\ \mathbf{I}_{M_k} & \mathbf{C}_{-k}^i \\ \mathbf{O}_{\overline{S}_k\times M_k} & \end{array}\right] \tag{10}$$

where

$$\mathbf{C}_{-k}^i = \text{Blockdiag}\left(\mathbf{X}_1^i, \ldots, \mathbf{X}_{k-1}^i, \mathbf{O}_{M_k\times M_k}, \mathbf{X}_{k+1}^i, \ldots, \mathbf{X}_K^i\right) \tag{11}$$

and where $\underline{S}_k = \sum_{q=1}^{k-1} M_q$ and $\overline{S}_k = \sum_{q=k+1}^{K} M_q$. It is noted that

$$\mathbf{R}_{\tilde{y}_k\tilde{y}_k}^i = \mathbf{C}_k^{i\,H}\mathbf{R}_{yy}\mathbf{C}_k^i\ . \tag{12}$$

[2] Since the algorithms envisaged in this paper are adaptive, short-term stationarity is sufficient.

[3] Here, we assume that $Q < M_k, \forall\ k \in \mathcal{K}$. If there exists a k for which $Q \geq M_k$, node k should transmit uncompressed observations of $\mathbf{y}_k$ to one other node, which will then treat $\mathbf{y}_k$ as part of its own observations.

Similarly to (2), $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}$ can be estimated as

$$\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k} \approx \frac{1}{N}\sum_{t=1}^{N} \widetilde{\mathbf{y}}^i_k[t]\widetilde{\mathbf{y}}^i_k[t]^H \ . \tag{13}$$

For later purpose, we also define the $Q \times Q$ matrix

$$\mathbf{D}^i_k = {\mathbf{X}^i_k}^H \mathbf{X}^i_k \tag{14}$$

and its square-root factorization

$$\mathbf{D}^i_k = {\mathbf{L}^i_k}^H \mathbf{L}^i_k \tag{15}$$

where $\mathbf{L}^i_k$ is a $Q \times Q$ matrix. It is noted that $\mathbf{L}^i_k$ is not unique and it can be computed by means of, e.g., a Cholesky factorization [10] or an EVD. Finally, we define the block-diagonal matrices

$$\mathbf{\Lambda}^i_k = \text{Blockdiag}\left(\mathbf{I}_{M_k}, \mathbf{L}^i_1, \ldots, \mathbf{L}^i_{k-1}, \mathbf{L}^i_{k+1}, \ldots, \mathbf{L}^i_K\right) \tag{16}$$

and its inverse

$$\mathbf{V}^i_k = \left(\mathbf{\Lambda}^i_k\right)^{-1} \ . \tag{17}$$

3.2. Algorithm derivation

We define the objective function

$$J\left(\mathbf{X}\right) = \text{Tr}\left\{\mathbf{X}^H \mathbf{R}_{yy} \mathbf{X}\right\} \tag{18}$$

where Tr$\{\cdot\}$ denotes the trace operator. Note that $\hat{\mathbf{X}}$ as defined in (4) maximizes (18) under the orthogonality constraint $\mathbf{X}^H\mathbf{X} = \mathbf{I}$. Consider the following alternating optimization (AO) procedure:

1. Set $i \leftarrow 0$, $q \leftarrow 1$, and $\mathbf{X}^0$ as a random $M \times Q$ matrix.
2. Choose $\mathbf{X}^{i+1}$ as a solution of:

$$\mathbf{X}^{i+1} \in \arg\max_{\mathbf{X}} J\left(\mathbf{X}\right) \tag{19}$$

$$\text{s.t.} \ \cdot \ \mathbf{X}^H\mathbf{X} = \mathbf{I} \tag{20}$$

$$\cdot \ \forall\, k \in \mathcal{K}\backslash\{q\} : \ \mathbf{X}_k \in \text{Range}\{\mathbf{X}^i_k\} \tag{21}$$

   where $\mathbf{X}_k$ is the k-th submatrix of $\mathbf{X}$ similarly defined as in (5), and where Range$\{\mathbf{X}^i_k\}$ denotes the subspace spanned by the columns of $\mathbf{X}^i_k$.
3. $i \leftarrow i+1$ and $q \leftarrow (q \mod K) + 1$.
4. Return to step 2.

Each iteration of the AO procedure increases the objective function $J\left(\mathbf{X}^i\right)$ in a monotonic fashion. Indeed, the constraint (21) changes in each iteration, allowing to update a particular submatrix of $\mathbf{X}$ freely (i.e., $\mathbf{X}_q$), while constraining the other submatrices to preserve their current column space. Despite the fact that this AO procedure is a centralized procedure requiring the full covariance matrix $\mathbf{R}_{yy}$, the particular form of the constraints (21) allows to execute it in a distributed fashion, which is explained next.

Notice that solving (19)-(21) is equivalent to solving

$$\widetilde{\mathbf{X}} \in \ \arg\max_{\widetilde{\mathbf{X}}} \text{Tr}\left\{\widetilde{\mathbf{X}}^H \mathbf{R}^i_{\tilde{y}_q\tilde{y}_q} \widetilde{\mathbf{X}}\right\} \tag{22}$$

$$\text{s.t.} \ \ \widetilde{\mathbf{X}}^H {\mathbf{C}^i_q}^H \mathbf{C}^i_q \widetilde{\mathbf{X}} = \mathbf{I} \tag{23}$$

and setting $\mathbf{X}^{i+1} = \mathbf{C}^i_q\widetilde{\mathbf{X}}$. Using the substitutions $\overline{\mathbf{X}} = \mathbf{\Lambda}^i_q\widetilde{\mathbf{X}}$ and $\overline{\mathbf{R}}^i_q = {\mathbf{V}^i_q}^H \mathbf{R}^i_{\tilde{y}_q\tilde{y}_q} \mathbf{V}^i_q$, and using the fact that ${\mathbf{C}^i_q}^H \mathbf{C}^i_q = {\mathbf{\Lambda}^i_q}^H \mathbf{\Lambda}^i_q$ and $\mathbf{\Lambda}^i_q\mathbf{V}^i_q = \mathbf{I}$, this is also equivalent to solving

$$\overline{\mathbf{X}} \in \ \arg\max_{\overline{\mathbf{X}}} \text{Tr}\left\{\overline{\mathbf{X}}^H \overline{\mathbf{R}}^i_q \overline{\mathbf{X}}\right\} \tag{24}$$

$$\text{s.t.} \ \ \overline{\mathbf{X}}^H\overline{\mathbf{X}} = \mathbf{I} \tag{25}$$

and setting $\mathbf{X}^{i+1} = \mathbf{C}^i_k\mathbf{V}^i_q\overline{\mathbf{X}}$. Note that this optimization problem can be solved by performing an EVD of $\overline{\mathbf{R}}^i_q$. Since $\mathbf{R}^i_{\tilde{y}_q\tilde{y}_q}$ can be estimated by node q based on (13), it can compute $\overline{\mathbf{R}}^i_q$ and its EVD (assuming $\mathbf{V}^i_q$ is known, which will require exchange of the $\mathbf{L}^i_k$'s between the nodes). The result can then be used to update the global $\mathbf{X}^i$ into $\mathbf{X}^{i+1}$. The DACMEE algorithm, as described in Table 1, iteratively performs these operations. As the DACMEE algorithm is then equivalent to the AO procedure, it will also result in a monotonic increase of Tr$\left\{\mathbf{X}^H\mathbf{R}_{yy}\mathbf{X}\right\}$ under the constraint $\mathbf{X}^H\mathbf{X} = \mathbf{I}$.

Table 1. The DACMEE algorithm in a fully connected WSN.

1. Set $i \leftarrow 0$, $q \leftarrow 1$, and initialize $\mathbf{X}^0_k, \forall\, k \in \mathcal{K}$, randomly.
2. Each node $k \in \mathcal{K}$ computes $\mathbf{D}^i_k = {\mathbf{X}^i_k}^H\mathbf{X}^i_k$ and its the square-root factorization $\mathbf{D}^i_k = {\mathbf{L}^i_k}^H\mathbf{L}^i_k$. The $Q \times Q$ matrix $\mathbf{L}^i_k$ is then broadcast to all other nodes.
3. Each node $k \in \mathcal{K}$ broadcasts N new compressed sensor signal observations $\mathbf{z}^i_k[iN+j] = {\mathbf{X}^i_k}^H \mathbf{y}_k[iN+j]$ (where $j = 1 \ldots N$) to all other nodes.
4. At node q:
   - Estimate $\mathbf{R}^i_{\tilde{y}_q\tilde{y}_q}$ with the N new observations of $\widetilde{\mathbf{y}}^i_q$ as in (13).
   - Construct the block-diagonal matrix $\mathbf{\Lambda}^i_q$ as defined in (16) and compute the inverse of each diagonal block to construct the block-diagonal matrix $\mathbf{V}^i_q = \left(\mathbf{\Lambda}^i_q\right)^{-1}$.
   - Compute the Q principal eigenvectors $\overline{\mathbf{X}}$ of the matrix $\overline{\mathbf{R}}^i_q = {\mathbf{V}^i_q}^H \mathbf{R}^i_{\tilde{y}_q\tilde{y}_q} \mathbf{V}^i_q$ (the Q columns of $\overline{\mathbf{X}}$ are sorted such that the corresponding eigenvalues are decreasing).
   - Set

$$\begin{bmatrix} \mathbf{X}^{i+1}_q \\ \mathbf{G}_{-q} \end{bmatrix} = \mathbf{V}^i_q\overline{\mathbf{X}} \tag{26}$$

$$\mathbf{D}^{i+1}_q = {\mathbf{X}^{i+1}_q}^H \mathbf{X}^{i+1}_q \ . \tag{27}$$

   - Compute the square-root factorization $\mathbf{D}^{i+1}_q = {\mathbf{L}^{i+1}_q}^H \mathbf{L}^{i+1}_q$.
   - Broadcast $\mathbf{L}^{i+1}_q$ and the matrix $\mathbf{G}_{-q}$ to all the other nodes.
5. Let $\mathbf{G}_{-q} = \left[\mathbf{G}^T_1 \ \ldots \ \mathbf{G}^T_{q-1} \ \mathbf{G}^T_{q+1} \ \ldots \ \mathbf{G}^T_K\right]^T$ where each partition consists of a $Q \times Q$ matrix. Each node $k \in \mathcal{K}\backslash\{q\}$ updates

$$\mathbf{L}^{i+1}_k = \mathbf{L}^i_k\mathbf{G}_k \tag{28}$$

$$\mathbf{X}^{i+1}_k = \mathbf{X}^i_k\mathbf{G}_k \ . \tag{29}$$

6. $i \leftarrow i+1$ and $q \leftarrow (q \mod K) + 1$.
7. Return to step 3.

It is noted that, in contrast to the AO procedure, the DACMEE algorithm is assumed to operate in an adaptive (time-recursive) context, and therefore all nodes collect and broadcast new observations in each iteration. The number of observations N that are collected

and broadcast in between the iterations (step 3) is chosen such that a sufficiently accurate estimate of $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}$ can be computed in step 4. Furthermore, since all nodes are assumed to act as a data sink for the Q-dimensional observations of $\mathbf{X}^{i\,H}\mathbf{y}$, the (compressed) observations of the z^i_k's are broadcast to all the nodes in the network, even though only one node q actually uses this data to update its local $\mathbf{X}^i_q$ in each iteration.

Remark I: It is noted that we have made two implicit assumptions to guarantee that the DACMEE algorithm in Table 1 is well-defined (which are usually satisfied in practice):

1. The matrix $\mathbf{\Lambda}^i_q$ has full rank, $\forall\, i \in \mathbb{N}$, with q being the updating node in iteration i.
2. The $Q+1$ largest eigenvalues of $\overline{\mathbf{R}}^i_q$ have multiplicity 1, where q is the updating node in iteration i.

This guarantees that $\mathbf{X}^{i+1}_q$ and $\mathbf{G}_{-q}$ are well-defined, i.e., there exists a unique $\overline{\mathbf{X}}$ in each iteration (up to a sign ambiguity). However, it is noted that these assumptions are made merely for the sake of an easy exposition, i.e., the degenerate case can easily be dealt with under some minor modifications to the algorithm.

Remark II: The extra data exchange for the $\mathbf{L}^{i+1}_q$ and $\mathbf{G}_{-q}$ matrices (step 5) is negligible compared to the intermediate exchange of the KN (compressed) observations of the $\mathbf{z}^i_k$'s. This also holds if the full matrix $\mathbf{X}^i$ were communicated to the other nodes (e.g., for KLT-based compression/decompression).

3.3. Convergence and optimality results

In the previous subsection, it has been explained that the DACMEE algorithm yields a monotonic increase of $J(\mathbf{X})$ (under the constraint $\mathbf{X}^H\mathbf{X} = \mathbf{I}$) and that $J(\hat{\mathbf{X}})$ is the corresponding maximum. However, this monotonic increase does not necessarily mean that the algorithm converges, let alone, that it converges to $\hat{\mathbf{X}}$. Therefore, we provide some stronger results on convergence and optimality[4].

Theorem 3.1 (proof omitted) $\mathbf{X}^* = \lim_{i\to\infty}\mathbf{X}^i$ exists, i.e., the DACMEE algorithm converges (for any initialization point).

Note that $\mathbf{X}^*$ is an equilibrium point of the DACMEE algorithm, i.e., if $\mathbf{X}^i = \mathbf{X}^*$ then $\forall\, j \geq i:\ \mathbf{X}^{j+1} = \mathbf{X}^j$. Although Theorem 3.1 does not make any statements about the optimality of $\mathbf{X}^*$, we can make a general statement about the set of equilibrium points:

Theorem 3.2 (proof omitted) Let $\mathcal{X}^*$ denote the set of all equilibrium points of the DACMEE algorithm. Every $\mathbf{X}^* \in \mathcal{X}^*$ can only have eigenvectors of $\mathbf{R}_{yy}$ in its columns. Furthermore, $\mathcal{X}^*$ always contains $\hat{\mathbf{X}}$, as defined in (4), which is the only stable equilibrium point under the DACMEE updates.

It is noted that Theorem 3.2 does not guarantee that $\mathcal{X}^*$ is a singleton, i.e., that $\hat{\mathbf{X}}$ is the only equilibrium point. Nevertheless, it is unlikely that multiple equilibria exist, since this requires existence of an eigenvector that maximizes the objective function over K different constraint sets defined by (20)-(21), $\forall\, q \in \mathcal{K}$. Moreover, even when such a suboptimal equilibrium point exists, it will be unstable. Due to inevitable estimation errors and numerical noise, we can safely assume that -in practice- the DACMEE algorithm will diverge from such an unstable equilibrium point. Since $\hat{\mathbf{X}}$ is the only stable equilibrium point, we can conclude that the DACMEE algorithm converges to the Q principal eigenvectors of $\mathbf{R}_{yy}$, i.e., $\mathbf{X}^\infty = \hat{\mathbf{X}}$.

[4] All theorems in this paper assume that the matrices $\mathbf{R}^i_{\tilde{y}_k\tilde{y}_k}, \forall k \in \mathcal{K}$, are estimated without errors. Small estimation errors may cause the algorithm to randomly move within a small neighborhood of the optimal solution.

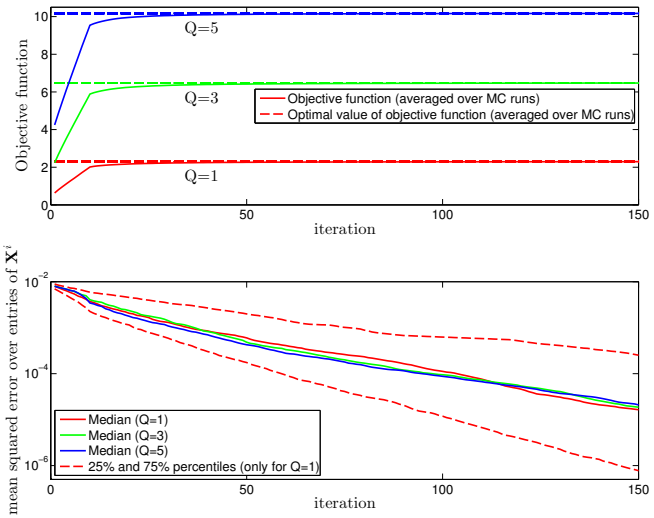

Fig. 1. Convergence properties of the DACMEE algorithm.

4. SIMULATIONS

In this section, we provide Monte-Carlo (MC) simulations of the DACMEE algorithm, and compare it with the centralized solution. In each MC run, a new scenario is created with $K = 10$ nodes, each collecting observations of a different 20-dimensional stochastic vector $\mathbf{y}_k, \forall\, k \in \mathcal{K}$s, where the observations of the stacked vector $\mathbf{y}$ are generated as

$$\mathbf{y}[t] = \mathbf{A}\mathbf{d}[t] + \mathbf{n}[t] \tag{30}$$

where $\mathbf{A}$ is a deterministic 200×20 matrix (independent from t) from which the entries are randomly drawn from a uniform distribution over the interval $[-0.5; 0.5]$, $\mathbf{d}[t]$ is an observation of a 20-dimensional stochastic vector from which the entries are independent and uniformly distributed over the interval $[-0.5; 0.5]$ and $\mathbf{n}[t]$ is an observation of a 200-dimensional stochastic vector from which the entries are independent and uniformly distributed over the interval $[-\sqrt{2}/4; \sqrt{2}/4]$ (modelling spatially uncorrelated sensor noise).

The upper part of Fig. 1 shows the monotonic increase of the objective function (18) over the different iterations of the DACMEE algorithm for different values of Q (averaged over 200 MC runs). We observe that, after a sufficient number of iterations, the algorithm always converges to the correct value. The bottom part of Fig. 1 shows the squared error over the entries of $\mathbf{X}^i$ compared to $\hat{\mathbf{X}}$, i.e.,

$$\frac{1}{MQ} Tr\left\{\left(\mathbf{X}^i - \hat{\mathbf{X}}\right)^H\left(\mathbf{X}^i - \hat{\mathbf{X}}\right)\right\}. \tag{31}$$

The plot shows the median (50% percentile) over the 200 MC runs for different values of Q. It is observed that Q has no significant influence on the convergence speed of the algorithm. For the case of $Q = 1$, the 25% and 75% percentile are also shown (the percentiles for $Q = 3$ and $Q = 5$ are similar but omitted here).

5. CONCLUSIONS

We have described a distributed adaptive (time-recursive) algorithm to estimate and track the eigenvectors corresponding to the Q largest or smallest eigenvalues of the global sensor signal covariance matrix in a fully connected sensor network. It has been demonstrated that the eigenvectors can be computed without the need to gather all the sensor observations in a fusion center. The theoretical results have been validated by means of numerical simulations.

6. REFERENCES

[1] S. Macua, P. Belanovic, and S. Zazo, "Consensus-based distributed principal component analysis in wireless sensor networks," in *International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, june 2010, pp. 1 –5.

[2] A. Bertrand and M. Moonen, "Consensus-based distributed total least squares estimation in ad hoc wireless sensor networks," *IEEE Trans. on Signal Processing*, vol. 59, no. 5, pp. 2320–2330, May 2011.

[3] ——, "Low-complexity distributed total least squares estimation in ad hoc sensor networks," *IEEE Transactions on Signal Processing*, vol. 60, pp. 4321–4333, Aug. 2012.

[4] Z.-J. Bai, R. H. Chan, and F. T. Luk, "Principal component analysis for distributed data sets with updating," in *Proceedings of the 6th international conference on Advanced Parallel Processing Technologies*, ser. APPT'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 471–483.

[5] A. Scaglione, R. Pagliari, and H. Krim, "The decentralized estimation of the sample covariance," in *Asilomar Conference on Signals, Systems and Computers*, oct. 2008, pp. 1722 –1726.

[6] L. Li, X. Li, A. Scaglione, and J. Manton, "Decentralized subspace tracking via gossiping," in *Distributed Computing in Sensor Systems*, ser. Lecture Notes in Computer Science, R. Rajaraman, T. Moscibroda, A. Dunkels, and A. Scaglione, Eds. Springer Berlin Heidelberg, 2010, vol. 6131, pp. 130–143.

[7] M. Gastpar, P. Dragotti, and M. Vetterli, "The distributed Karhunen-Loève transform," in *IEEE Workshop on Multimedia Signal Processing*, dec. 2002, pp. 57 – 60.

[8] Y.-A. Le Borgne, S. Raybaud, and G. Bontempi, "Distributed principal component analysis for wireless sensor networks," *Sensors*, vol. 8, no. 8, pp. 4821–4850, 2008.

[9] I. Markovsky and S. Van Huffel, "Overview of total least-squares methods," *Signal Processing*, vol. 87, no. 10, pp. 2283 – 2302, 2007, special Section: Total Least Squares and Errors-in-Variables Modeling.

[10] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore: The Johns Hopkins University Press, 1996.