DECENTRALIZED COOPERATIVE DOA TRACKING USING NON-HERMITIAN GENERALIZED EIGENDECOMPOSITION

Wassim Suleiman, Marius Pesavento, Abdelhak M. Zoubir

Institute of Telecommunications, Technische Universität Darmstadt
(suleiman, zoubir)@spg.tu-darmstadt.de, pesavento@nt.tu-darmstadt.de

ABSTRACT

The problem of direction-of-arrival (DOA) estimation using partly calibrated arrays composed of multiple identically oriented subarrays is considered. The subarrays are assumed to possess the shift-invariance property which is exploited to develop a distributed search-free DOA estimation algorithm that is based on the generalized eigendecomposition (GED) of a pair of covariance matrices. We propose a fully decentralized adaptive algorithm which tracks the generalized eigenvalues (GEVs) of a non-Hermitian pair of covariance matrices, from which the DOAs are estimated. Moreover, to enforce the amplitude property of the nominal source GEDs, we propose a suitable measurement weighting scheme. We demonstrate the estimation performance of our algorithm with simulations and confirm that our algorithm is able to identify more sources than each subarray individually can.

Index Terms— partly calibrated arrays, cooperative DOA tracking, decentralized generalized eigendecomposition

1. INTRODUCTION

DOA estimation using sensor arrays is essential for many applications such as radar, sonar, underwater surveillance, and seismic exploration [1, 2]. For these applications sensor arrays with large apertures and large sensor numbers are attractive as they offer high angular resolution and are able to identify a large number of sources. Due to high calibration cost, the large array is split into smaller fully calibrated subarrays with unknown displacement between the subarrays. Thus, they are referred to as partly calibrated arrays. Centralized DOA estimation algorithms using partly calibrated arrays are introduced in [3–5]. These algorithms possess high resolution capabilities and are able to identify a large number of sources. However, centralized algorithms require a powerful fusion center (FC) and a large communication bandwidth at the subarrays to communicate the measurements to the FC. In [6–9], decentralized subspace-based DOA estimation algorithms were introduced in which each subarray locally computes sufficient statistics and communicates them to a fusion center (FC). The aforementioned algorithms rely on the assumption that each subarray can individually identify all the sources. Furthermore, due to transmit power limitations, subarrays at a distance far from the FC are not able to transfer their local data on a direct link to the FC. In these situations, multi-hop communication is required, which, however, is associated with many difficulties, such as the need for discovering and maintaining routing information and dealing with node failures, to name a few [10].

Based on the ESPRIT algorithm [11], in [12] and [13], we introduced the D-ESPRIT and L-ESPRIT methods, respectively, for DOA estimation in partly calibrated arrays, where L-ESPRIT is superior to D-ESPRIT in terms of the associated communication cost. The D-ESPRIT and L-ESPRIT methods eliminate the need for a FC by the use of the averaging consensus (AC) algorithm [14]. The AC algorithm calculates averages of initial scalar measurements that are distributed among all the subarrays, using only local communication between neighboring subarrays, consequently, eliminating multi-hop communication. The D-ESPRIT and L-ESPRIT methods perform batch processing, i.e., in both algorithms, first the individual subarrays collect and store measurements, then, they perform decentralized DOA estimation which requires computational power and communication between the subarrays. This results in an unbalanced usage of subarray resources (memory, bandwidth and processing power). Moreover, the D-ESPRIT and L-ESPRIT methods comprise a Least Square estimation step which requires a communication cost that scales quadratically with the number of sources [12]. In this paper, we propose a decentralized DOA estimation algorithm which overcomes these shortcomings.

This paper is organised as follows. In Section 2 the DOA estimation problem is cast as the problem of finding the GEVs of a pair of covariance matrices, where one matrix is non-Hermitian, as presented in [15]. In Section 3, based on the natural power method [16], which tracks the eigenvectors of a Hermitian matrix, we introduce a decentralized algorithm for tracking the GEVs of a non-Hermitian pair of matrices. We also introduce a scheme for weighting the measurements to enforce constraints on the amplitude of the sought GEVs. In Section 4, we analyze the communication and memory costs of our algorithm and in Section 5 we evaluate its performance using simulations. Finally, we conclude the paper in Section 6.

We remark that, similar as in the D-ESPRIT and L-ESPRIT methods, cooperation between the subarrays enables the presented algorithm to identify more DOAs than identifiable by the individual subarrays. Moreover, although our algorithm is introduced for DOA estimation using non-Hermitian matrix pair, it can be applied to other GED problems including Hermitian problems, such as, classification [17, p. 186] and blind source separation [18].

In this paper, $(\cdot)^*$, $(\cdot)^T$, $(\cdot)^H$, $\odot$, $\boldsymbol{I}_i$ and $\mathbf{0}_{i,j}$ denote complex conjugate, transpose, conjugate transpose, element-wise multiplication, the $i \times i$ identity matrix, zero matrix of size $i \times j$, respectively. Diagonal or block diagonal matrices, the set of complex numbers, the argument of a complex number, and the cardinality of a set are denote as $\mathrm{diag}[\cdot]$, $\mathbb{C}$, $\arg[\cdot]$ and $\mathrm{card}[\cdot]$, respectively.

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

2. SIGNAL MODEL

Consider an array composed of K identically oriented uniform linear subarrays. The kth subarray is composed of M_k sensors separated by distance d, measured in half-wavelength. The array includes in total $M \triangleq \sum_{k=1}^{K} M_k$ antennas. Displacements between the subarrays are considered to be unknown. The first $M_k - 1$ sensor and last $M_k - 1$ sensors at the kth subarray are denoted as the upper and lower groups [12], respectively. Correspondingly, upper and lower selection matrices are defined as $\overline{\boldsymbol{J}}_k \triangleq [\boldsymbol{I}_{M_k-1}, \boldsymbol{0}_{M_k-1\times 1}]$ and $\underline{\boldsymbol{J}}_k \triangleq [\boldsymbol{0}_{M_k-1\times 1}, \boldsymbol{I}_{M_k-1}]$, respectively. Note that the upper and lower sensor groups can be regarded as shifted versions of each other. This property of the array is very essential for estimating the DOAs using ESPRIT [11] or the GED algorithm in [15]. We remark that our model can be generalized to any subarray geometry which maintains this shift invariance property. Consider L independent narrowband far-field sources whose signals impinge on the planar array from directions $\theta_1, \ldots, \theta_L$. The measurement vector of the kth subarray at time t is

$$\boldsymbol{x}_k(t) = \boldsymbol{A}_k \boldsymbol{s}(t) + \boldsymbol{n}_k(t), \tag{1}$$

where $\boldsymbol{s}(t) \in \mathbb{C}^{L\times 1}$ is the source signal vector, $\boldsymbol{A}_k \in \mathbb{C}^{M_k \times L}$ is the kth subarray steering matrix [12] and $\boldsymbol{n}_k(t) \in \mathbb{C}^{M_k\times 1}$ is the zero mean circular white Gaussian sensor noise with variance σ^2. Let $\overline{\boldsymbol{x}}_k(t) \triangleq \overline{\boldsymbol{J}}_k \boldsymbol{x}_k(t)$ and $\underline{\boldsymbol{x}}_k(t) \triangleq \underline{\boldsymbol{J}}_k \boldsymbol{x}_k(t)$. The overall measurement model can be formulated as

$$\boldsymbol{x}(t) = \boldsymbol{A}\boldsymbol{s}(t) + \boldsymbol{n}(t) \tag{2}$$

where $\boldsymbol{x}(t) \triangleq [\boldsymbol{x}_1^T(t), \ldots, \boldsymbol{x}_K^T(t)]^T$ and $\boldsymbol{A} \triangleq [\boldsymbol{A}_1^T, \ldots, \boldsymbol{A}_K^T]^T$ and $\boldsymbol{n}(t) \triangleq [\boldsymbol{n}_1^T(t), \ldots, \boldsymbol{n}_K^T(t)]^T$. Correspondingly, we define, $\overline{\boldsymbol{x}}(t) \triangleq [\overline{\boldsymbol{x}}_1^T(t), \ldots, \overline{\boldsymbol{x}}_K^T(t)]^T = \overline{\boldsymbol{J}}\boldsymbol{x}(t)$ and $\underline{\boldsymbol{x}}(t) \triangleq [\underline{\boldsymbol{x}}_1^T(t), \ldots, \underline{\boldsymbol{x}}_K^T(t)]^T = \underline{\boldsymbol{J}}\boldsymbol{x}(t)$, where $\overline{\boldsymbol{J}} \triangleq \mathrm{diag}[\overline{\boldsymbol{J}}_1, \ldots, \overline{\boldsymbol{J}}_K]$ and $\underline{\boldsymbol{J}} \triangleq \mathrm{diag}[\underline{\boldsymbol{J}}_1, \ldots, \underline{\boldsymbol{J}}_K]$.

In analogy to [15], we define two $(M-K)\times(M-K)$ covariance matrices $\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}} \triangleq \mathbb{E}[\overline{\boldsymbol{x}}(t)\overline{\boldsymbol{x}}^H(t)]$ and $\boldsymbol{R}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}} \triangleq \mathbb{E}[\underline{\boldsymbol{x}}(t)\overline{\boldsymbol{x}}^H(t)]$. Let $\gamma_1, \ldots, \gamma_L \in \mathbb{C}$ be the L GEVs of the matrix pair $(\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}, \boldsymbol{R}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}})$ which have the largest amplitudes, then,

$$\theta_l = \sin^{-1}\big(\arg[\gamma_l]/(d\pi)\big), \tag{3}$$

for $l = 1, \ldots, L$, for details see [15].

3. THE DECENTRALIZED NON-HERMITIAN GED

In this section, we propose an online adaptive algorithm, which tracks the generalized eigenvalues (GEVs) of the non-Hermitian matrix pair $(\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}, \boldsymbol{R}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}})$ for each measurement vector. First, we introduce the algorithm in a centralized manner then we show how it can be implemented in a fully decentralized fashion using averaging consensus (AC).

3.1. The Centralized Non-Hermitian GED

The lth GEV [19, p. 233] of the matrix pair $(\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}, \boldsymbol{R}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}})$ and its corresponding right generalized eigenvector $\boldsymbol{u}_l$ are defined as

$$\boldsymbol{R}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}}\boldsymbol{u}_l = \gamma_l \boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}\boldsymbol{u}_l. \tag{4}$$

By multiplying Eq. (4) with $\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}^{-1}$, the GED is reduced to an eigendecomposition of the form

$$\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}^{-1}\boldsymbol{R}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}}\boldsymbol{u}_l = \gamma_l \boldsymbol{u}_l. \tag{5}$$

In very large sensor networks, the dimensions of $\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}$ are very large, consequently, computing its inverse is impractical. Thus, iterative methods for finding γ_l and $\boldsymbol{u}_l$ from Eq. (5) are sought. In these iterative methods, multiplying a vector with $\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}^{-1}$ is achieved by iteratively solving a system of linear equations, see [19, ch. 8] for example. In decentralized implementations, the iteration procedure results in a large undesired communication cost. To overcome this problem, we propose to approximate the matrix $\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}$, which has only L dominant eigenvalues corresponding to the L DOAs [11], with

$$\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}} \approx \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^H, \tag{6}$$

where $\boldsymbol{\Lambda} \triangleq \mathrm{diag}[\lambda_1, \ldots, \lambda_L]$ and $\boldsymbol{W} \triangleq [\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L]$ are the matrices containing the largest L eigenvalues of $\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}$ and their corresponding eigenvectors, respectively. Substituting Eq. (6) in Eq. (5) yields

$$\boldsymbol{W}\boldsymbol{\Lambda}^{-1}\boldsymbol{W}^H\boldsymbol{R}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}}\boldsymbol{u}_l = \gamma_l \boldsymbol{u}_l. \tag{7}$$

We rewrite Eq. (7) in matrix form as

$$\boldsymbol{W}\boldsymbol{\Lambda}^{-1}\boldsymbol{W}^H\boldsymbol{R}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Gamma}, \tag{8}$$

where $\boldsymbol{\Gamma} \triangleq \mathrm{diag}[\gamma_1, \ldots, \gamma_L]$ and $\boldsymbol{U} \triangleq [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_L]$.

The sample estimates of $\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}$ and $\boldsymbol{R}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}}$ can be defined as a rank one update at each time t, as follows

$$\begin{aligned}\hat{\boldsymbol{R}}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}(t) &= \alpha\hat{\boldsymbol{R}}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}(t-1) + \overline{\boldsymbol{x}}(t)\,\overline{\boldsymbol{x}}^H(t)\\ \hat{\boldsymbol{R}}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}}(t) &= \alpha\hat{\boldsymbol{R}}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}}(t-1) + \underline{\boldsymbol{x}}(t)\,\overline{\boldsymbol{x}}^H(t),\end{aligned} \tag{9}$$

where $0 \le \alpha \le 1$ is a forgetting factor [16]. Let $\hat{\boldsymbol{\Lambda}}(t), \hat{\boldsymbol{W}}(t), \hat{\boldsymbol{\Gamma}}(t)$ and $\hat{\boldsymbol{U}}(t)$ be the sample estimates of $\boldsymbol{\Lambda}(t), \boldsymbol{W}(t), \boldsymbol{\Gamma}(t)$ and $\boldsymbol{U}(t)$, respectively, at time t.

Our algorithm tracks $\hat{\boldsymbol{\Lambda}}(t), \hat{\boldsymbol{W}}(t), \hat{\boldsymbol{\Gamma}}(t)$ and $\hat{\boldsymbol{U}}(t)$ and updates them according to the newly acquired measurement vectors $\overline{\boldsymbol{x}}(t)$ and $\underline{\boldsymbol{x}}(t)$ at time t. The power method [19, p. 51] is used for this update since even for non-Hermitian matrices, the power iteration converges to the eigenvalue with the largest amplitude [20].

Tracking $\hat{\boldsymbol{W}}(t)$ is achieved by using the approximate simultaneous power method, which is proposed in [16] under the name "natural power method" (NP2). At time t, the NP2 computes $\hat{\boldsymbol{W}}(t)$ as follows

$$\hat{\boldsymbol{W}}(t) = \boldsymbol{Y}(t)\boldsymbol{C}^{-1/2}(t), \tag{10}$$

where $\boldsymbol{Y}(t) \triangleq \hat{\boldsymbol{R}}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}(t)\hat{\boldsymbol{W}}(t-1)$ and $\boldsymbol{C}(t) \triangleq \boldsymbol{Y}^H(t)\boldsymbol{Y}(t)$. Substituting Eq. (9) in the definition of $\boldsymbol{Y}(t)$ yields

$$\boldsymbol{Y}(t) = \alpha\boldsymbol{Y}(t-1) + \overline{\boldsymbol{x}}(t)\overline{\boldsymbol{y}}^H(t) \tag{11}$$

where $\overline{\boldsymbol{y}}(t) \triangleq \hat{\boldsymbol{W}}^H(t-1)\overline{\boldsymbol{x}}(t)$ and the approximation $\hat{\boldsymbol{W}}(t-1) \approx \hat{\boldsymbol{W}}(t-2)$ is used [16]. Moreover, the matrix $\boldsymbol{C}(t)$ can be written as

$$\boldsymbol{C}(t) = \alpha^2\boldsymbol{C}(t-1) + \boldsymbol{c}(t)\overline{\boldsymbol{y}}^H(t) + \overline{\boldsymbol{y}}(t)\boldsymbol{c}^H(t) + \eta_{\overline{\boldsymbol{x}}}(t)\overline{\boldsymbol{y}}(t)\overline{\boldsymbol{y}}^H(t), \tag{12}$$

where $\boldsymbol{c}(t) \triangleq \alpha\boldsymbol{Y}^H(t-1)\overline{\boldsymbol{x}}(t) = \alpha\big(\boldsymbol{C}^{1/2}(t-1)\big)^H\overline{\boldsymbol{y}}(t)$ and $\eta_{\overline{\boldsymbol{x}}}(t) = \overline{\boldsymbol{x}}^H(t)\overline{\boldsymbol{x}}(t)$. Equations (10)–(12) represent the iteration of the NP2 as introduced in [16]. Note that although this iteration updates the eigenvectors of the matrix $\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}$, the update of its eigenvalues $\boldsymbol{\Lambda}(t)$ is not considered. Thus, we propose to achieve the update of $\boldsymbol{\Lambda}(t)$ as follows.

Substituting Eq. (9) in the definition of the eigenvalues $\hat{\boldsymbol{\Lambda}}(t) \triangleq \hat{\boldsymbol{W}}^H(t)\hat{\boldsymbol{R}}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}(t)\hat{\boldsymbol{W}}(t)$ produces

$$\hat{\boldsymbol{\Lambda}}(t) = \alpha\hat{\boldsymbol{W}}^H(t)\hat{\boldsymbol{R}}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}(t-1)\hat{\boldsymbol{W}}(t) + \overline{\boldsymbol{y}}(t)\overline{\boldsymbol{y}}^H(t). \tag{13}$$

In analogy to [16], we use the approximation $\hat{\boldsymbol{W}}(t) \approx \hat{\boldsymbol{W}}(t-1)$, in Eq. (13) which yields

$$\hat{\boldsymbol{\Lambda}}(t) = \alpha\hat{\boldsymbol{\Lambda}}(t-1) + \text{diag}[\overline{\boldsymbol{y}}(t) \odot \overline{\boldsymbol{y}}^*(t)], \quad (14)$$

where the off-diagonal elements of $\overline{\boldsymbol{y}}(t)\overline{\boldsymbol{y}}^H(t)$ are zeros since both matrices $\hat{\boldsymbol{\Lambda}}(t)$ and $\hat{\boldsymbol{\Lambda}}(t-1)$ are diagonal. We remark that, by using (10)–(12) and (14), we are able to track and update both $\boldsymbol{\Lambda}(t)$ and $\boldsymbol{W}(t)$. In the following, we apply the NP2 and the proposed eigenvalues update in Eq. (14) to achieve the GED of Eq. (8), i.e., to track $\boldsymbol{\gamma}(t)$ and $\boldsymbol{U}(t)$.

In analogy to Eq. (10), applying the NP2 to the eigendecomposition in Eq. (8) results in the following iteration

$$\hat{\boldsymbol{U}}(t) = \boldsymbol{Z}(t)\boldsymbol{G}^{-1/2}(t) \quad (15)$$

where $\boldsymbol{G}(t) \triangleq \boldsymbol{Z}^H(t)\boldsymbol{Z}(t)$ and $\boldsymbol{Z}(t) \triangleq \hat{\boldsymbol{W}}(t-1)\hat{\boldsymbol{\Lambda}}^{-1}(t-1)\hat{\boldsymbol{W}}^H(t-1)\hat{\boldsymbol{R}}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}}(t)\hat{\boldsymbol{U}}(t-1)$. Substituting Eq. (9) in the definition of $\boldsymbol{Z}(t)$ yields

$$\boldsymbol{Z}(t) = \alpha\boldsymbol{Z}(t-1) + \boldsymbol{h}(t)\boldsymbol{z}^H(t), \quad (16)$$

where $\boldsymbol{z}(t) \triangleq \hat{\boldsymbol{U}}^H(t-1)\overline{\boldsymbol{x}}(t)$ and $\boldsymbol{h}(t) \triangleq \hat{\boldsymbol{W}}(t-1)\hat{\boldsymbol{\Lambda}}^{-1}(t-1)\underline{\boldsymbol{y}}(t)$ and $\underline{\boldsymbol{y}}(t) \triangleq \hat{\boldsymbol{W}}^H(t-1)\underline{\boldsymbol{x}}(t)$ and the approximations $\hat{\boldsymbol{W}}(t-2) \approx \hat{\boldsymbol{W}}(t-1)$ and $\hat{\boldsymbol{U}}(t-2) \approx \hat{\boldsymbol{U}}(t-1)$ are used.

The matrix $\boldsymbol{G}(t)$ can be rewritten as

$$\boldsymbol{G}(t) = \alpha^2\boldsymbol{G}(t-1) + \boldsymbol{g}(t)\boldsymbol{z}^H(t) + \boldsymbol{z}(t)\boldsymbol{g}^H(t) + \eta_{\boldsymbol{h}}(t)\boldsymbol{z}(t)\boldsymbol{z}^H(t), \quad (17)$$

where $\boldsymbol{g}(t) \triangleq \alpha\boldsymbol{Z}^H(t-1)\boldsymbol{h}(t) = \alpha(\boldsymbol{G}^{1/2}(t-1))^H\boldsymbol{f}(t)$ and $\boldsymbol{f}(t) \triangleq \hat{\boldsymbol{U}}^H(t-1)\boldsymbol{h}(t)$ and $\eta_{\boldsymbol{h}}(t) \triangleq \boldsymbol{h}^H(t)\boldsymbol{h}(t)$. Similar to our proposition in Eq. (14) the GEVs can be updated as

$$\hat{\boldsymbol{\Gamma}}(t) = \alpha\hat{\boldsymbol{\Gamma}}(t-1) + \text{diag}[\boldsymbol{f}(t) \odot \boldsymbol{z}^*(t)]. \quad (18)$$

Using Equations (15)–(18), we are able to track the generalized eigenvalues and eigenvectors of the matrix pair $(\boldsymbol{R}_{\overline{\boldsymbol{x}}\overline{\boldsymbol{x}}}, \boldsymbol{R}_{\underline{\boldsymbol{x}}\overline{\boldsymbol{x}}})$.

The centralized tracking of the GEVs is summarized in Algorithm 1. Note that we rearranged the steps and divided them into two parts. Part I contains all the operations which require communication between the subarrays in the decentralized implementation that will be introduced later. Part II contains local updates which are carried out at the subarrays and do not require communication between the subarrays.

Since the GEVs corresponding to true sources have unity amplitude [15], we introduce a weighting factor $\beta(t)$, in Part II of Algorithm 1, to impose this property of the GEVs. The multiplication factor $\beta(t)$ penalizes the measurements $\overline{\boldsymbol{x}}(t)$ and $\underline{\boldsymbol{x}}(t)$ which produce GEVs with non-unity amplitude as follows. First the iteration of Algorithm 1 is computed with $\beta(t) = 1$ and the GEVs are computed. Then, we set

$$\beta(t) = \exp\big(-|\log(\frac{1}{L}\sum_{l=1}^{L}|\hat{\gamma}_l(t)|)|\big) \quad (19)$$

and we repeat only Part II of Algorithm 1. Since Part II of Algorithm 1 contains only local updates, the communication cost required for imposing unity amplitude on the GEVs is kept unchanged. We remark that the derivation of the steps in Part II is achieved by using the penalized measurements $\beta(t)\overline{\boldsymbol{x}}(t)$ and $\beta(t)\underline{\boldsymbol{x}}(t)$ instead of $\overline{\boldsymbol{x}}(t)$ and $\underline{\boldsymbol{x}}(t)$, respectively.

In the next section, we show how Algorithm 1 can be implemented in decentralized fashion, using the AC algorithm.

Algorithm 1 Generalized Eigendecomposition

Step 0: Init $\hat{\boldsymbol{W}}(0)$ and $\hat{\boldsymbol{U}}(0)$ at random and orthogonalize them. Set $\boldsymbol{Y}(0), \boldsymbol{Z}(0), \boldsymbol{C}(0), \boldsymbol{G}(0), \hat{\boldsymbol{\Lambda}}(0)$ and $\hat{\boldsymbol{\Gamma}}(0)$ to zero.
for each input $\overline{\boldsymbol{x}}(t)$ and $\underline{\boldsymbol{x}}(t)$ **do**
Part I: Network computation (AC)
Step 1 (AC1): $\overline{\boldsymbol{y}}(t) \leftarrow \hat{\boldsymbol{W}}^H(t-1)\overline{\boldsymbol{x}}(t)$
Step 2 (AC1): $\underline{\boldsymbol{y}}(t) \leftarrow \hat{\boldsymbol{W}}^H(t-1)\underline{\boldsymbol{x}}(t)$
Step 3 (AC1): $\eta_{\overline{\boldsymbol{x}}}(t) \leftarrow \overline{\boldsymbol{x}}^H(t)\overline{\boldsymbol{x}}(t)$
Step 4 (AC1): $\boldsymbol{z}(t) \leftarrow \hat{\boldsymbol{U}}^H(t-1)\overline{\boldsymbol{x}}(t)$
Step 5: $\boldsymbol{h}(t) \leftarrow \hat{\boldsymbol{W}}(t-1)\hat{\boldsymbol{\Lambda}}^{-1}(t-1)\underline{\boldsymbol{y}}(t)$
Step 6 (AC2): $\eta_{\boldsymbol{h}}(t) \leftarrow \boldsymbol{h}^H(t)\boldsymbol{h}(t)$
Step 7 (AC2): $\boldsymbol{f}(t) \leftarrow \hat{\boldsymbol{U}}^H(t-1)\boldsymbol{h}(t)$
Part II: Node computation (local update)
Step 8: $\boldsymbol{Y}(t) \leftarrow \alpha\boldsymbol{Y}(t-1) + \beta^2(t)\,\overline{\boldsymbol{x}}(t)\overline{\boldsymbol{y}}^H(t)$
Step 9: $\boldsymbol{c}(t) \leftarrow \alpha\big(\boldsymbol{C}^{1/2}(t-1)\big)^H\overline{\boldsymbol{y}}(t)$
Step 10: $\boldsymbol{C}(t) \leftarrow \alpha^2\boldsymbol{C}(t-1) + \beta^2(t)\boldsymbol{c}(t)\overline{\boldsymbol{y}}^H(t) + \beta^2(t)\overline{\boldsymbol{y}}(t)\boldsymbol{c}^H(t) + \beta^4(t)\eta_{\overline{\boldsymbol{x}}}(t)\overline{\boldsymbol{y}}(t)\overline{\boldsymbol{y}}^H(t)$
Step 11: $\hat{\boldsymbol{W}}(t) \leftarrow \boldsymbol{Y}(t)\boldsymbol{C}^{-1/2}(t)$
Step 12: $\hat{\boldsymbol{\Lambda}}(t) = \alpha\hat{\boldsymbol{\Lambda}}(t-1) + \beta^2(t)\text{diag}[\overline{\boldsymbol{y}}(t) \odot \overline{\boldsymbol{y}}^*(t)]$
Step 13: $\boldsymbol{Z}(t) \leftarrow \alpha\boldsymbol{Z}(t-1) + \beta^2(t)\boldsymbol{h}(t)\boldsymbol{z}^H(t)$
Step 14: $\boldsymbol{g}(t) \leftarrow \alpha(\boldsymbol{G}^{1/2}(t-1))^H\boldsymbol{f}(t)$
Step 15: $\boldsymbol{G}(t) \leftarrow \alpha^2\boldsymbol{G}(t-1) + \beta^2(t)\boldsymbol{g}(t)\boldsymbol{z}^H(t) + \beta^2(t)\boldsymbol{z}(t)\boldsymbol{g}^H(t) + \beta^4(t)\eta_{\boldsymbol{h}}(t)\boldsymbol{z}(t)\boldsymbol{z}^H(t)$
Step 16: $\hat{\boldsymbol{U}}(t) \leftarrow \boldsymbol{Z}(t)\boldsymbol{G}^{-1/2}(t)$
Step 17: $\hat{\boldsymbol{\Gamma}}(t) \leftarrow \alpha\hat{\boldsymbol{\Gamma}}(t-1) + \beta^2(t)\text{diag}[\boldsymbol{f}(t) \odot \boldsymbol{z}^*(t)]$
Step 18: DOA estimation using Eq. (3).
end for

3.2. Averaging Consensus

Assume that the kth subarray stores a scalar $\chi_k(0)$. In the AC algorithm, the average $\overline{\chi}(0) \triangleq \frac{1}{K}\sum_{k=1}^{K}\chi_k(0)$, can be computed in a decentralized fashion using only local communications between neighboring subarrays [14]. The AC algorithm uses the following iteration

$$\tilde{\overline{\chi}}_k(i) = \tilde{\overline{\chi}}_k(i-1) + \sum_{j\in\mathcal{N}_k}\omega_{j,k}(\tilde{\overline{\chi}}_j(i-1) - \tilde{\overline{\chi}}_k(i-1)), \quad (20)$$

where $\omega_{j,k}$ is the weight associated with the communication link between the jth and the kth subarrays and $\mathcal{N}_k$ is the set of subarrays in the vicinity of the kth subarray. The AC algorithm is initialized with $\tilde{\overline{\chi}}_k(0) = \chi_k(0)$. We use the notation $\tilde{(\cdot)}$ to denote that the resulting average estimate is available at all subarrays.

Different alternatives for choosing the weights $\omega_{j,k}$ exist, e.g., we can choose

$$\omega_{j,k} = \begin{cases} 1/\max(\text{card}[\mathcal{N}_j], \text{card}[\mathcal{N}_k]), & \text{if } j \in \mathcal{N}_k \\ 0, & \text{otherwise;} \end{cases} \quad (21)$$

refer to [14] for more details.

3.3. The Decentralized GED

In our proposed decentralized implementation each subarray stores locally the part of the variables $\hat{\boldsymbol{W}}(t), \hat{\boldsymbol{U}}(t), \boldsymbol{Y}(t), \boldsymbol{Z}(t)$ and $\boldsymbol{h}(t)$ which corresponds to its measurements. Thus we

partition these variables as follows, $\hat{\boldsymbol{W}}(t) \triangleq [\hat{\boldsymbol{W}}_1^T(t) \ , \ldots , \hat{\boldsymbol{W}}_K^T(t)]^T$, where the kth matrix block $\hat{\boldsymbol{W}}_k(t)$ is stored locally at the kth subarray. Note that in this partition, each subarray stores a part of the lth estimated eigenvector $\hat{\boldsymbol{w}}_l(t) \triangleq [\hat{\boldsymbol{w}}_{l,1}^T(t), \ldots, \hat{\boldsymbol{w}}_{l,K}^T(t)]^T$. The same partition is assumed for $\hat{\boldsymbol{U}}(t), \boldsymbol{Y}(t), \boldsymbol{Z}(t)$ and $\boldsymbol{h}(t)$. Moreover, in our proposed decentralized implementation, using the AC algorithm, the kth subarray maintains a local copy of the following variables $\tilde{\overline{\boldsymbol{y}}}_k(t)$, $\tilde{\boldsymbol{c}}_k(t)$, $\tilde{\underline{\boldsymbol{y}}}_k(t)$, $\tilde{\eta}_{\overline{\boldsymbol{x}}}(t)$, $\tilde{\boldsymbol{z}}_k(t)$, $\tilde{\eta}_{\boldsymbol{h}}(t)$, $\tilde{\boldsymbol{f}}_k(t)$, $\tilde{\boldsymbol{g}}_k(t)$, $\tilde{\hat{\boldsymbol{\Lambda}}}_k(t)$, $\tilde{\hat{\boldsymbol{\Gamma}}}_k(t)$, $\tilde{\beta}_k(t)$, $\tilde{\boldsymbol{C}}_k(t)$ and $\tilde{\boldsymbol{G}}_k(t)$, where $\tilde{(\cdot)}$ is used as in Section 3.2.

In Algorithm 1, we have arranged all operations which require the AC algorithm in Part I. These operations are of two types. The first type is matrix vector multiplication (MVM) and includes steps 1, 2, 4 and 7. The second type is vector vector multiplication (VVM) and it includes steps 3 and 6. Step 1 which is a MVM can be rewritten as

$$\overline{\boldsymbol{y}}(t) = [\hat{\boldsymbol{w}}_1^H(t-1)\overline{\boldsymbol{x}}(t), \ldots, \hat{\boldsymbol{w}}_L^H(t-1)\overline{\boldsymbol{x}}(t)]^T. \qquad (22)$$

The lth entry of the vector $\overline{\boldsymbol{y}}(t)$ can be rewritten as an average of K scalars which are distributed over the K subarrays as follows [12, 21]

$$\hat{\boldsymbol{w}}_l^H(t-1)\overline{\boldsymbol{x}}(t) = K\Big(\frac{1}{K}\sum_{k=1}^{K}\xi_{l,k}(t)\Big), \qquad (23)$$

where the scalar $\xi_{l,k}(t) \triangleq \hat{\boldsymbol{w}}_{l,k}^H(t-1)\overline{\boldsymbol{x}}_k(t)$ is computed locally at the kth subarray. In our decentralized implementation, the AC algorithm, introduced in Section 3.2, is used to compute this average such that all subarrays maintain access to this average. Thus, using L AC operations the kth subarray computes a local estimate of $\overline{\boldsymbol{y}}(t)$ which we denoted earlier as $\tilde{\overline{\boldsymbol{y}}}_k(t)$. The remaining MVM operations are achieved in similar manner. Also the VVMs in steps 3 and 6 are carried out as in Eq. (23).

Note that Step 6 and all steps of Part II of Algorithm 1 are local steps, where the kth subarray computes either its local copy of the variables $\tilde{\hat{\boldsymbol{\Lambda}}}_k(t), \tilde{\hat{\boldsymbol{\Gamma}}}_k(t), \tilde{\boldsymbol{C}}_k(t)$, $\tilde{\boldsymbol{G}}_k(t)$, $\beta_k(t)$ and the L DOAs, or it computes the part of the variables $\hat{\boldsymbol{W}}(t), \hat{\boldsymbol{U}}(t), \boldsymbol{Y}(t), \boldsymbol{Z}(t)$ and $\boldsymbol{h}(t)$ which correspond to its measurements.

4. COST ANALYSIS

In Algorithm 1, the AC algorithm is used for MVM in steps 1, 2, 4 and 7. Each of these MVMs requires L AC operations. The AC algorithm is also used for VVM in steps 3 and 6. Thus the communication cost associated with the Algorithm at each iteration is $4L + 2 = \mathcal{O}(L)$ AC operations.

The memory required at each subarray is found by directly summing the memory cost of variables or part of variables which the subarray stores. This cost is $4LM_k + 2L^2 = \mathcal{O}(LM_k) + \mathcal{O}(L^2)$ floating-point numbers, where variables costing less than L^2 or M_kL floating-point numbers are neglected.

The steps of Algorithm 1 which are marked as AC1 do not depend on each other and are carried out in parallel using $4L + 1$ parallel AC algorithm. Note that running parallel AC algorithms minimizes the latency and the communication overhead of our proposed algorithm. The same applies to the steps which are marked as AC2.

5. SIMULATION RESULTS

We consider an array composed of $K = 30$ uniform linear subarrays, where each subarray consists of $M_k = 2$ sensors. The locations of the subarrays are generated at random using a uniform distribution over the rectangle whose lower-left and upper-right corners are (0, 0) and (10, 7), respectively. Two nodes are connected if the distance between them is less than 2.5.

In the first simulation, the signals of $L = 3$ equal-powered stationary sources impinge onto the array from directions $-15°$, $-3°$ and $25°$ with SNR $= 10$ dB. For each AC operation, 6 iterations of Eq. (20) are carried out, using the weighting scheme from Eq. (21). The forgetting factor is taken to be $\alpha = 0.99$. Fig. 1 illustrates the DOA estimates obtained from one subarray for $t = 1, \ldots, 3000$. Note that after $t = 500$ our algorithm is able to resolve the three DOAs within a reasonable accuracy.

In the second simulation, two equal-powered moving sources with SNR $= 10$ dB are considered. The direction of the sources are changing linearly with time from $\theta_1 = 40°$ and $\theta_2 = 0°$ at $t = 0$ to $\theta_1 = 0°$ and $\theta_2 = 40°$ at $t = 3000$. The parameter setup of the AC algorithm is taken as in the first simulation. We set the forgetting factor to $\alpha = 0.95$. Fig. 2 and Fig. 3 display the estimated DOAs, without weighting and with weighting, respectively, at one subarray for $t = 1, \ldots, 3000$. In both figures, our Algorithm is able to track the DOAs after $t = 200$. However, when the angular separation between the two sources is small, which correspond to the region around $t = 1500$ in Fig. 2 and Fig. 3, our algorithm is not able to resolve the two sources, and a noisy GEV appears causing errors. In Fig. 2, the errors around $t = 1500$ are larger compared with Fig. 3 where we penalize measurements producing GEVs with amplitude different than unity.

Note that in both simulations, each subarray can only estimate one DOA, since each subarray has only 2 sensors, however, our algorithm is able to identify and track more than one source.

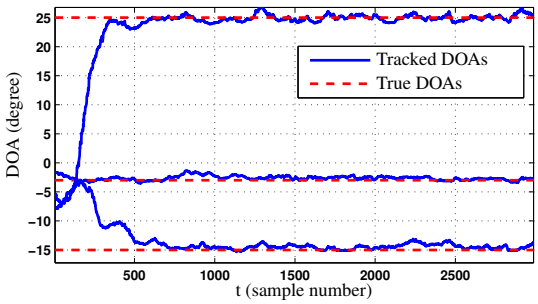

Fig. 1. Stationary sources

6. CONCLUSION

In this paper, we proposed a decentralized DOA tracking algorithm, where subarrays cooperate with each other to track more sources than individually resolvable by each subarray. We proposed a weighting scheme to benefit from the fact that the DOAs correspond to GEVs with unity amplitude. Our algorithm is not restricted to DOA estimation but can also be

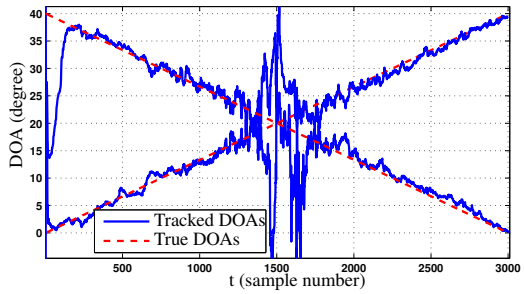

Fig. 2. Tracking moving sources

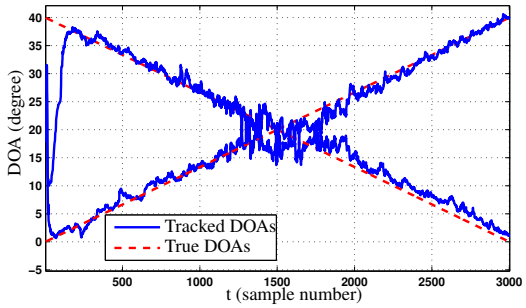

Fig. 3. Tracking moving sources with weighting

used for any GED of two covariance matrices even if one of them is non-Hermitian.

7. ACKNOWLEDGEMENT

The project ADEL acknowledges the financial support of the Seventh Framework Programme for Research of the European Commission under grant number: 619647.

REFERENCES

[1] H. L. Van Trees, *Detection, Estimation, and Modulation Theory - Part IV: Optimum Array Processing*, Wiley-Interscience, 2002.

[2] H. Krim and M. Viberg, "Two decades of array signal processing research: the parametric approach," *IEEE Signal Processing Magazine*, vol. 13, no. 4, pp. 67–94, July 1996.

[3] A.L. Swindlehurst, P. Stoica, and M. Jansson, "Exploiting arrays with multiple invariances using MUSIC and MODE," *IEEE Transactions on Signal Processing*, vol. 49, no. 11, pp. 2511–2521, Nov 2001.

[4] M. Pesavento, A. B. Gershman, and K. M. Wong, "Direction finding in partly calibrated sensor arrays composed of multiple subarrays," *IEEE Transactions on Signal Processing*, vol. 50, no. 9, pp. 2103–2115, Sept. 2002.

[5] P. Parvazi, M. Pesavento, and A. B. Gershman, "Direction-of-arrival estimation and array calibration for partly-calibrated arrays," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 2552–2555.

[6] M. Wax and T. Kailath, "Decentralized processing in sensor arrays," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 33, no. 5, pp. 1123–1129, Oct. 1985.

[7] T. Söderström and P. Stoica, "Statistical analysis of decentralized MUSIC," *Springer Circuits, Systems and Signal Processing*, vol. 11, no. 4, pp. 443–454, Dec. 1992.

[8] P. Stoica, A. Nehorai, and T. Söderström, "Decentralized array processing using the MODE algorithm," *Circuits, Systems and Signal Processing*, vol. 14, no. 1, pp. 17–38, 1995.

[9] W. Suleiman and P. Parvazi, "Search-free decentralized direction-of-arrival estimation using common roots for non-coherent partly calibrated arrays," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 2292–2296.

[10] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Transactions on Information Theory*, vol. 52, no. 6, pp. 2508–2530, 2006.

[11] R. Roy and T. Kailath, "ESPRIT-estimation of signal parameters via rotational invariance techniques," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 7, pp. 984–995, Nov. 1989.

[12] W. Suleiman, M. Pesavento, and A. M. Zoubir, "Decentralized direction finding using partly calibrated arrays," in *European signal processing conference (EUSIPCO)*, Sept. 2013, pp. 1–5.

[13] W. Suleiman, P. Parvazi, M. Pesavento, and A. M. Zoubir, "Decentralized direction finding using Lanczos method," in *IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, June 2014, pp. 9–12.

[14] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, vol. 53, no. 1, pp. 65–78, 2004.

[15] H. Ouibrahim, *A generalized approach to direction finding*, Ph.D. thesis, Syracuse University, 1986.

[16] Y. Hua, Y. Xiang, T. Chen, K. Abed-Meraim, and Y. Miao, "A new look at the power method for fast subspace tracking," *Digital Signal Processing*, vol. 9, no. 4, pp. 297–314, 1999.

[17] C. Bishop, *Pattern recognition and machine learning*, vol. 1, springer New York, 2006.

[18] C. Chang, Z. Ding, S. Yau, and F. Chan, "A matrix-pencil approach to blind separation of colored nonstationary signals," *IEEE Transactions on Signal Processing*, vol. 48, no. 3, pp. 900–907, Mar. 2000.

[19] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the solution of algebraic eigenvalue problems: a practical guide*, vol. 11, Society for Industrial and Applied Mathematics (SIAM), 2000.

[20] GW Stewart, "Simultaneous iteration for computing invariant subspaces of non-hermitian matrices," *Numerische Mathematik*, vol. 25, no. 2, pp. 123–136, 1976.

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