This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

Distributed algorithms for array signal processing

Po-Chih Chen, *Student member, IEEE* and P. P. Vaidyanathan, *Life Fellow, IEEE*

***Abstract*—Distributed or decentralized estimation of covariance, and distributed principal component analysis have been introduced and studied in the signal processing community in recent years, and applications in array processing have been indicated in some detail. Inspired by these, this paper provides a detailed development of several distributed algorithms for array processing. New distributed algorithms are proposed for DOA estimation methods like root-MUSIC, *total* least squares-ESPRIT, and FOCUSS. Other contributions include distributed design of the Capon beamformer from data, and distributed implementation of the spatial smoothing method for coherent sources. A distributed implementation of a recently proposed beamspace method called the *convolutional* beamspace (CBS) is also proposed. The proposed algorithms are fully distributed – an average consensus (AC) is used to avoid the need for a fusion center. The algorithms are based on a recently reported finite-time version of AC which converges to the exact solution in a finite number of iterations. Numerical examples are given throughout the paper to show the effectiveness of the proposed algorithms.**

***Index Terms*—Distributed or decentralized DOA estimation, Capon beamforming, convolutional beamspace, root-MUSIC, ESPRIT, FOCUSS, spatial smoothing.**

I. Introduction

BEAMFORMING and direction-of-arrival (DOA) estimation are two important areas of sensor array processing [1]–[5]. Popular algorithms include the Capon method for beamforming [1] and MUSIC [2], root-MUSIC [3], ESPRIT [4], and FOCUSS [6], [7] for DOA estimation. Traditionally, these algorithms require data collection and centralized computation at a fusion center. However, following the pioneering work of Scaglione, et al. [8], distributed algorithms for DOA estimation and beamforming have gained more research interest. In these algorithms, the sensor array is partitioned into subarrays. The data in subarray i is available only to the processor in that subarray, and between the processors there is some minimal exchange of intermediate results, in order to implement an average consensus. Based on such local computations and limited data exchange between processors, the goal is to perform usual array processing tasks such as DOA estimation and beamforming. Thus, such arrays work without the help of a central processor or fusion center. The communication bottleneck that is present in the case of large arrays with a central processor is thus mitigated in these decentralized (or distributed) systems. A detailed discussion of the relevance and advantages of such distributed processors can be found in [9] and references therein.

An excellent overview of distributed implementations of principal component methods is presented in [9], with a mention of applications in array processing as well. In the above papers for distributed array processing, a network gossiping protocol called average consensus (AC) [10]–[12] is extensively used as a backbone algorithm to avoid the need of fusion centers. AC is a method for computing the average of some values stored at all the subarrays. Subspace-based methods for DOA estimation, including MUSIC [2], root-MUSIC [3], and ESPRIT [4], require the computation of the eigenvalue decomposition (EVD) of the covariance matrix of the array output. To compute the EVD in a fully distributed manner, a distributed power method is proposed in [8]. In [13], Suleiman, et al. propose a distributed algorithm for ESPRIT based on least-squares estimates (LS-ESPRIT), although the total least squares (TLS) ESPRIT is not considered. All the above are AC-based methods and apply to any network structure. Bertrand, et al. [14] propose a non-AC-based method that uses an alternating optimization procedure to estimate covariance matrix eigenvectors. The communication cost is reduced by sending array data projected onto lower dimensional subspaces. However, this method applies only to fully connected networks or networks with a tree topology. Using a similar idea, a distributed DOA estimation method which applies only to fully connected networks is presented in [15]. Partially distributed algorithms for MUSIC and root-MUSIC are introduced in earlier literature [16] and [17], but fusion centers are still required therein. We focus on AC-based methods in this paper as it applies to any network structure. The main goal of this paper is to show how to use the distributed power method and AC to implement several important array processing algorithms, but not to compare different kinds of distributed eigenvector estimation methods or AC methods.

Importantly, there are two families of AC methods, *asymptotic convergence* [10] and the recently introduced *finite-time convergence* methods [11], [12]. Asymptotic AC is used for the distributed DOA estimation algorithms in previous works [8], [9], [13], [18]. In these methods, one uses only finite but sufficiently many iterations to approximate asymptotic behaviors, so additional estimation errors arise from the use of asymptotic AC. This kind of error is analyzed in [18] for distributed LS-ESPRIT. By contrast, finite-time AC offers exact convergence in a finite number of iterations, so no additional estimation errors are introduced. Hence, we choose to use finite-time AC in this paper, and this is why our distributed algorithms can achieve the same performance as the centralized counterparts. Finite-time AC can be applied without any limitations, so they are readily applicable to the previous works [8], [9], [13], [18] as well. Exact convergence is guaranteed in finite iterations as long as the subarray network is connected [11]. The finite-time AC methods [11], [12] are based on the idea of linear graph filters. When we are allowed to use a large enough filter order, i.e., a sufficient number of AC iterations, these finite-time AC

This work was supported in parts by the ONR grant N00014-18-1-2390, the NSF grant CCF-1712633, and the California Institute of Technology.

The authors are with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125, USA (email: pochih@caltech.edu; ppvnath@systems.caltech.edu).

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

methods already yield zero MSE due to exact convergence. We consider this scenario in the paper. If one wants to use fewer iterations while sacrificing some MSE, a new finite-time AC method [19] may be considered. The new method extends the idea to nonlinear graph filters and designs filter coefficients using a learning framework of graph convolutional neural networks.

Although there are several inspiring papers on distributed algorithms for array processing, distributed algorithms have not yet been reported for a number of well-known methods for DOA estimation and beamforming. In this paper we develop distributed algorithms for DOA estimation methods such as root-MUSIC, and total least squares (TLS) ESPRIT which is known to be more accurate than LS-ESPRIT [13]. We also derive distributed versions of the Capon beamformer and the well-known FOCUSS method for sparse-solver based DOA estimation. The above DOA estimation methods are classical methods, and we only consider these. More recent methods like grid-based Lasso [20] and grid-less atomic norm minimization [21] are left as future work. Besides, *beamspace* processing is a well-known technique in array processing [5], [22]–[32], and in Sec. IV we propose distributed algorithms for a beamspace method called convolutional beamspace (CBS) recently introduced in [31], [32]. We will show that distributed DOA estimation algorithms, including root-MUSIC, TLS-ESPRIT, and FOCUSS, can be applied either directly to the original array domain, called *element-space*, or in series with a beamspace method like CBS. We also propose a distributed algorithm for spatial smoothing [33], which is a technique used for DOA estimation when there are coherent or correlated sources.

All the proposed algorithms are *fully distributed* in that a fusion center is not required. The novelties of the proposed algorithms mainly lie in finding a way to implementing the algorithm so that all the data exchange among subarrays can be realized using AC. This is achieved by transforming problems at hand into steps where computing the average of some values across the network is the only step that involves data exchange among subarrays. In particular, we often transform the problems into a series of linear operations on the involved distributed variables, such as array outputs and signal eigenvectors. For instance, in Capon beamforming, it would not be easy if one tries to compute the inverse of the covariance matrix directly when the array output is distributed. The novelty of our distributed Capon beamforming is to propose to use the conjugate gradient method [34] to replace the inversion of the covariance matrix with linear operations on array outputs. Likewise, only linear operations on array outputs are needed in distributed FOCUSS. As another example, in root-MUSIC, it would not be easy if one tries to explicitly compute $\hat{\mathbf{E}}_{\rm s}\hat{\mathbf{E}}_{\rm s}^H$ when the eigenvectors are distributed, where the columns of $\hat{\mathbf{E}}_{\rm s}$ are the estimates of the signal eigenvectors. The novelty of our distributed root-MUSIC is to propose to avoid explicit computation of $\hat{\mathbf{E}}_{\rm s}\hat{\mathbf{E}}_{\rm s}^H$ and use the Aberth method [35] for rooting polynomials in root-MUSIC to ensure that only linear operations on signal eigenvectors are involved. That is, many of the algorithms in their original form require operations other than weighted averages of the involved data, and we show how to transform those operations into a series of weighted averages. In order to put the new methods in the right context, this paper includes short reviews of all important techniques which form the backbone of the new methods. Moreover, the existing and new, centralized and distributed methods are presented in a unified framework. In this sense the paper is self contained and has some tutorial value as well.

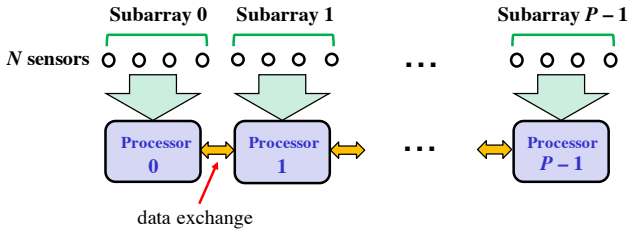


Fig. 1. Schematic of distributed array processing. The N-sensor linear array is divided into P subarrays. The sensor data from subarray i is directly available only to processor i, corresponding to node i in a communication network modeled by an undirected graph $\mathcal{G}$. Between the processors there is some minimal exchange of intermediate results, in order to implement an average consensus.

The system model used throughout the paper is shown in Fig. 1. This is a network composed of P nodes, each of which is a Q-sensor linear subarray [8], [13]. For ease of presentation, we assume that each subarray has the same number of sensors, but many algorithms in this paper can be readily extended to subarrays with different numbers of sensors. The sensor data from subarray i is directly available only to a local processor at node i. Between the processors there is some minimal exchange of intermediate results, in order to implement an average consensus. The communication network is modeled by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of the P nodes, and $\mathcal{E}$ is the set of edges. Each node represents a set of sensors and the edges represent the communication links. If there is an edge between two nodes, then two way communication is allowed between these nodes (for average consensus and so forth). The P subarrays, which do not have overlapping sensors, collectively form a linear array with $N = PQ$ sensors. That is, all the subarrays are on the same vertical positions. The unit sensor spacing is $\lambda/2$, and monochromatic plane waves of wavelength λ arrive from D directions. Assume all the subarrays are located not too far away, and they receive the same set of source amplitudes. Let $\mathbf{x}_p \in \mathbb{C}^Q$ be the output of the pth subarray. Then the array output is

$$\mathbf{x} = [\mathbf{x}_0^T\ \mathbf{x}_1^T \cdots \mathbf{x}_{P-1}^T]^T = \mathbf{A}\mathbf{c} + \mathbf{e}, \tag{1}$$

where $\mathbf{c}$ contains source amplitudes c_i, $\mathbf{e}$ contains additive noise terms, and $\mathbf{A} = [\mathbf{a}_N(\omega_1)\ \mathbf{a}_N(\omega_2) \cdots \mathbf{a}_N(\omega_D)]$ with

$$\mathbf{a}_N(\omega) = [e^{j\omega z_0}\ e^{j\omega z_1} \cdots e^{j\omega z_{N-1}}]^T. \tag{2}$$

Here $z_i \in \mathbb{Z}$ is the ith sensor location. Without loss of generality, assume $z_0 = 0$. In (2), $\omega = \pi \sin\theta$, with DOA $\theta \in [-\pi/2, \pi/2)$ measured from the normal to the line of array. We assume $\mathrm{E}[\mathbf{c}] = \mathbf{0}$, $\mathrm{E}[|c_i|^2] = p_i$, $\mathrm{E}[\mathbf{e}] = \mathbf{0}$, $\mathrm{E}[\mathbf{e}\mathbf{e}^H] = \sigma_e^2\mathbf{I}$, and $\mathrm{E}[\mathbf{c}\mathbf{e}^H] = \mathbf{0}$. In this paper, we consider in general non-uniform linear arrays, but some of the proposed algorithms only apply to uniform linear arrays (ULAs), and we will mention it whenever it is the case.

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

Paper outline: The distributed power method and average consensus are reviewed in Sec. II. The proposed distributed algorithms for Capon beamforming, root-MUSIC, TLS-ESPRIT, and FOCUSS are introduced in Sec. III. Then distributed convolutional beamspace methods, along with a variation called the "robust Capon beamspace filter" are introduced in Sec. IV. Distributed spatial smoothing is described in Sec. V. Sec. VI concludes the paper.

Notations: Boldfaced capital letters denote matrices and boldfaced lowercase letters are reserved for column vectors. We use $(\cdot)^*$, $(\cdot)^T$, $(\cdot)^H$, and $(\cdot)^+$ to denote complex conjugate, transpose, conjugate transpose, and pseudoinverse, respectively. The identity matrix is denoted by $\mathbf{I}$, and $\mathrm{E}[\cdot]$ is the expectation operator.

II. Review of Distributed Power Method and Average Consensus

In this section, the distributed power method [8] and average consensus (AC) [10]–[12] are reviewed. The distributed power method is the first step for subspace-based DOA estimation algorithms as it estimates the eigenvectors of the covariance matrix of the array output. AC is a backbone subroutine for the distributed power method and other algorithms proposed in this paper.

Subspace-based methods for DOA estimation, including MUSIC [2], root-MUSIC [3], and ESPRIT [4], require the computation of the eigenvalue decomposition (EVD) of the covariance matrix $\mathbf{R}_{\mathbf{xx}}$ of the array output $\mathbf{x}$. In practice, we use K snapshots to get the covariance estimate

$$\widehat{\mathbf{R}}_{\mathbf{xx}} = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}[k]\mathbf{x}^H[k]. \tag{3}$$

To compute the EVD of $\widehat{\mathbf{R}}_{\mathbf{xx}}$ in a fully distributed manner, a distributed power method was proposed in [8]. The method uses average consensus (AC) [10]–[12] as a subroutine to compute the average of some values stored at all the subarrays. For average consensus, one can use methods which have either *asymptotic convergence* [10] or *finite-time convergence* [11], [12]. We propose to use finite-time AC in this paper since it offers exact convergence in a finite number of iterations. A numerical example in Sec. III-A also shows that finite-time AC performs better than asymptotic AC. As shown below, as long as $\mathcal{G}$ is a connected graph, exact convergence can be achieved in at most $P-1$ iterations [11].

A. Average Consensus

The aim of AC is to compute the average of some initial scalar values $u_p(0)$ stored at nodes $p = 0, \ldots, P-1$. Nodes communicate with their neighbors and update their values through distributed linear iterations of the form

$$u_p(t+1) = W_{pp}(t)u_p(t) + \sum_{i\in\mathcal{N}_p} W_{pi}(t)u_i(t), \tag{4}$$

where $W_{pi}(t)$ is the weight on $u_i(t)$ at node p for iteration t, and $\mathcal{N}_p = \{i \mid \{p,i\} \in \mathcal{E}\}$ is the set of neighbors of node p. The weights $W_{pi}(t)$ are in general complex-valued and time-varying, though they are restricted to be time independent in some papers [8], [10]. We can write (4) in the vector form

$$\mathbf{u}(t+1) = \mathbf{W}(t)\mathbf{u}(t). \tag{5}$$

The goal of finite-time AC is to achieve

$$\mathbf{u}(I_{\mathrm{ac}}) = \mathbf{1}\mathbf{1}^T\mathbf{u}(0)/P \tag{6}$$

after a finite number of iterations I_{ac}, while the goal of asymptotic AC is to achieve $\lim_{t\to\infty}\mathbf{u}(t) = \mathbf{1}\mathbf{1}^T\mathbf{u}(0)/P$. We propose to use finite-time AC in this paper. In [11], it was shown that if $\mathcal{G}$ is a connected graph, then there exist weight matrices $\mathbf{W}(0), \ldots, \mathbf{W}(I_{\mathrm{ac}}-1)$ with $I_{\mathrm{ac}} \le P-1$ such that (6) holds. Such weights are not unique, but one way to get a feasible solution is to start from the Laplacian $\mathbf{L}$ of $\mathcal{G}$. Suppose the distinct eigenvalues of $\mathbf{L}$ are $\lambda_1, \ldots, \lambda_R$, where $R \le P$. Since $\mathcal{G}$ is connected, $\mathbf{L}$ must have the simple eigenvalue 0 [36]. Without loss of generality, assume $\lambda_R = 0$. It is shown in [11] that if we take

$$\mathbf{W}(0) = \frac{(-1)^{R-1}}{\lambda_1\lambda_2\cdots\lambda_{R-1}}(\mathbf{L} - \lambda_1\mathbf{I}) \tag{7}$$

and

$$\mathbf{W}(t) = \mathbf{L} - \lambda_{t+1}\mathbf{I} \tag{8}$$

for $t = 1, \ldots, R-2$, the iteration (5) converges after $I_{\mathrm{ac}} = R-1 \le P-1$ steps.

B. Distributed Power Method

Now we summarize the distributed power method [8]. We first consider the derivation of the first eigenvector of $\widehat{\mathbf{R}}_{\mathbf{xx}}$. Let $\mathbf{e}_1(0) \in \mathbb{C}^N$ be an initial random vector. Suppose all eigenvalues are distinct. Then the power iterations [37]

$$\mathbf{e}_1(n+1) = \widehat{\mathbf{R}}_{\mathbf{xx}}\mathbf{e}_1(n) \tag{9}$$

will converge to the first eigenvector as $n \to \infty$. Let $\mathbf{e}_1(n) = [\mathbf{e}_{1,0}^T(n)\ \ \mathbf{e}_{1,1}^T(n)\cdots\mathbf{e}_{1,P-1}^T(n)]^T$ with each $\mathbf{e}_{1,p}(n) \in \mathbb{C}^Q$ stored locally at node p. Using (1) and (3), we can rewrite (9) as

$$\mathbf{e}_1(n+1) = \frac{1}{K}\sum_{k=1}^{K}\mathbf{x}[k]\sum_{p=0}^{P-1}\mathbf{x}_p^H[k]\mathbf{e}_{1,p}(n) \tag{10}$$

$$= \frac{P}{K}\sum_{k=1}^{K}\mathbf{x}[k]\,\mathrm{AC}_p(\mathbf{x}_p^H[k]\mathbf{e}_{1,p}(n)), \tag{11}$$

where $\mathrm{AC}_p(u_p)$ denotes the average consensus of the scalar values u_p with enough iterations so that the output is the exact average of u_p's. Hence, by first computing $b[k] = \mathrm{AC}_p(\mathbf{x}_p^H[k]\mathbf{e}_{1,p}(n))$, node p can locally compute

$$\mathbf{e}_{1,p}(n+1) = \frac{P}{K}\sum_{k=1}^{K}\mathbf{x}_p[k]b[k]. \tag{12}$$

Note that here we use the notation $b[k]$ instead of $b_p[k]$ because each node obtains a copy of the exact average, but one should understand that node p is using its own copy of $b[k]$ to compute (12). Suppose we run (9) for I_{pm} iterations. Then we do normalization $\hat{\mathbf{e}}_1 = \mathbf{e}_1(I_{\mathrm{pm}})/\|\mathbf{e}_1(I_{\mathrm{pm}})\|$ to obtain our final

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

estimate $\hat{\mathbf{e}}_1$ of the normalized first eigenvector. The norm $\|\mathbf{e}_1(I_{\rm pm})\|$ can also be computed via AC since

$$\|\mathbf{e}_1(I_{\rm pm})\|^2 = P \cdot \mathrm{AC}_p(\mathbf{e}_{1,p}^H(I_{\rm pm})\mathbf{e}_{1,p}(I_{\rm pm})). \quad (13)$$

To derive the qth eigenvector $\hat{\mathbf{e}}_q$ of $\widehat{\mathbf{R}}_{\mathbf{xx}}$, we note that $\hat{\mathbf{e}}_q$ is the eigenvector corresponding to the largest eigenvalue of $(\mathbf{I}-\sum_{i=1}^{q-1}\hat{\mathbf{e}}_i\hat{\mathbf{e}}_i^H)\widehat{\mathbf{R}}_{\mathbf{xx}}$, where $\hat{\mathbf{e}}_i$ is the ith eigenvector of $\widehat{\mathbf{R}}_{\mathbf{xx}}$. Thus, it can be obtained by running the power iterations

$$\mathbf{e}_q(n+1) = \left(\mathbf{I} - \sum_{i=1}^{q-1}\hat{\mathbf{e}}_i\hat{\mathbf{e}}_i^H\right)\widehat{\mathbf{R}}_{\mathbf{xx}}\mathbf{e}_q(n). \quad (14)$$

Note that $\bar{\mathbf{e}}_q(n) \triangleq \widehat{\mathbf{R}}_{\mathbf{xx}}\mathbf{e}_q(n)$ can be computed in a manner similar to (11). Then,

$$\mathbf{e}_q(n+1) = \bar{\mathbf{e}}_q(n) - P\sum_{i=1}^{q-1}\hat{\mathbf{e}}_i\,\mathrm{AC}_p(\hat{\mathbf{e}}_{i,p}^H\bar{\mathbf{e}}_{q,p}(n)), \quad (15)$$

where $\hat{\mathbf{e}}_{i,p}, \bar{\mathbf{e}}_{q,p}(n) \in \mathbb{C}^Q$ are subvectors of $\hat{\mathbf{e}}_i$ and $\bar{\mathbf{e}}_q(n)$, respectively, corresponding to node p. Hence, by first computing $f_{i,q} = \mathrm{AC}_p(\hat{\mathbf{e}}_{i,p}^H\bar{\mathbf{e}}_{q,p}(n))$, node p can locally compute

$$\mathbf{e}_{q,p}(n+1) = \bar{\mathbf{e}}_{q,p}(n) - P\sum_{i=1}^{q-1}\hat{\mathbf{e}}_{i,p}f_{i,q}, \quad (16)$$

which is the subvector of $\mathbf{e}_q(n+1)$ corresponding to node p. Finally, after $I_{\rm pm}$ iterations, we do normalization $\hat{\mathbf{e}}_q = \mathbf{e}_q(I_{\rm pm})/\|\mathbf{e}_q(I_{\rm pm})\|$ to obtain our final estimate $\hat{\mathbf{e}}_q$ of the normalized qth eigenvector. The norm $\|\mathbf{e}_q(I_{\rm pm})\|$ can be computed in a manner similar to (13). The total communication cost per edge for estimating D eigenvectors is $O(D(D+K)I_{\rm ac}I_{\rm pm}) \approx O(DKI_{\rm ac}I_{\rm pm})$ if $D \ll K$. The cost is mainly dominated by computing $\bar{\mathbf{e}}_q(n)$ and (15). For comparison, the communication cost of the method in [14] applied to a fully connected network is $O(DKI_{\rm ao})$, where $I_{\rm ao}$ is the number of iterations for alternating optimization. When the distributed power method is applied to a fully connected network, $I_{\rm ac} = 1$, so the communication cost is $O(DKI_{\rm pm})$. The communication costs of the two methods have the same dependence on D and K. We use distributed power method in this paper as it applies to any network structure.

The existence of $\hat{\mathbf{e}}_i$, $i < q$ in (14) implies that we can start the power iterations for $\hat{\mathbf{e}}_q$ only after the first $q-1$ eigenvectors are obtained. That is, the eigenvectors are updated sequentially. Alternatively, we can replace each $\hat{\mathbf{e}}_i$ by $\mathbf{e}_i(n+1)$, so the power iterations for all eigenvectors can be done in parallel. This can reduce time complexity. However, the modified method typically requires more iterations to compute each eigenvector, as shown later in Fig. 5. The total computational and communication costs are thus increased. Hence, we use only the original method in all other examples in this paper.

III. Distributed DOA Estimation and Beamforming

In this section, we propose distributed algorithms for Capon beamforming [1], root-MUSIC [3], ESPRIT [4] based on total least-squares estimates (TLS-ESPRIT), and FOCUSS [6], [7]. (Note that LS-ESPRIT has been reported [13] and analyzed [18].) The distributed DOA estimation algorithms, including root-MUSIC, TLS-ESPRIT, and FOCUSS, can be applied either directly to element-space or in series with a beamspace method, such as CBS presented in Sec. IV. In the following, we first show how to do distributed Capon beamforming in Sec. III-A. Distributed root-MUSIC, ESPRIT, and FOCUSS are presented in Secs. III-B, III-C, and III-D, respectively. FOCUSS is a method to obtain sparse solutions to underdetermined equations, so distributed FOCUSS has applications much broader than distributed DOA estimation. The communication, computation, and storage costs of the proposed distributed methods are compared to centralized methods in Sec. III-E.

A. Distributed Capon Beamforming

In traditional beamforming, we consider a linear array output

$$\mathbf{x} = c_0\mathbf{a}_N(\omega_0) + \mathbf{u}, \quad (17)$$

where $\mathbf{u} = \mathbf{A}\mathbf{c}+\mathbf{e}$ is the interference plus noise. The quantity ω_0 represents the "look direction," i.e., the direction in which we want to point the beam, c_0 is the amplitude of the signal coming from that direction, and $\mathbf{A}$, $\mathbf{c}$, $\mathbf{e}$ are as defined in (1), with $\omega_1, \ldots, \omega_D$ representing D interferer directions. The output of the beamformer can be expressed as

$$z_{\rm BF} = \mathbf{h}^H\mathbf{x} \quad (18)$$

where $\mathbf{h} = [h(0)\cdots h(N-1)]^T$ is a complex weighting vector. The output signal-to-interference-plus-noise ratio (SINR) of the beamformer is defined as

$$\mathrm{SINR} = \frac{\mathrm{E}[|c_0\mathbf{h}^H\mathbf{a}_N(\omega_0)|^2]}{\mathrm{E}[|\mathbf{h}^H\mathbf{u}|^2]} = \frac{p_0|\mathbf{h}^H\mathbf{a}_N(\omega_0)|^2}{\mathbf{h}^H\mathbf{R}_{\mathbf{uu}}\mathbf{h}}, \quad (19)$$

where $p_0 = \mathrm{E}[|c_0|^2]$ and $\mathbf{R}_{\mathbf{uu}} = \mathrm{E}[\mathbf{u}\mathbf{u}^H]$. When the signal is uncorrelated with the interference, the Capon beamformer [1], [5], which is the solution to the optimization problem

$$\begin{aligned} \min_{\mathbf{h}} \quad & \mathbf{h}^H\mathbf{R}_{\mathbf{xx}}\mathbf{h} \\ \text{subject to} \quad & \mathbf{h}^H\mathbf{a}_N(\omega_0) = 1, \end{aligned} \quad (20)$$

maximizes the SINR, where $\mathbf{R}_{\mathbf{xx}} = \mathrm{E}[\mathbf{x}\mathbf{x}^H]$. The solution to this problem is given by

$$\mathbf{h} = c\cdot\mathbf{R}_{\mathbf{xx}}^{-1}\mathbf{a}_N(\omega_0), \quad (21)$$

where $c = 1/(\mathbf{a}_N^H(\omega_0)\mathbf{R}_{\mathbf{xx}}^{-1}\mathbf{a}_N(\omega_0))$. In practice, $\mathbf{R}_{\mathbf{xx}}$ is replaced by its estimate $\widehat{\mathbf{R}}_{\mathbf{xx}}$ as in (3).

Now we show how to compute the Capon beamformer in our distributed setting, where the array output $\mathbf{x} = [\mathbf{x}_0^T\ \mathbf{x}_1^T\cdots\mathbf{x}_{P-1}^T]^T$ with $\mathbf{x}_p \in \mathbb{C}^Q$ stored locally at node p. The crucial step is to compute the inverse of the covariance estimate (3) appearing in (21). Instead of directly computing this inverse, we propose to use the conjugate gradient (CG) method [34] to compute

$$\mathbf{w} = \widehat{\mathbf{R}}_{\mathbf{xx}}^{-1}\mathbf{a}_N(\omega_0). \quad (22)$$

Equivalently, we want to solve the system of linear equations

$$\widehat{\mathbf{R}}_{\mathbf{xx}}\mathbf{w} = \mathbf{a}_N(\omega_0), \quad (23)$$

where $\widehat{\mathbf{R}}_{\mathbf{xx}} = \widehat{\mathbf{R}}_{\mathbf{xx}}^H$ is assumed to be positive definite (it is generally true if $K \geq N$). The CG method is a method for

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

Algorithm 1 Conjugate gradient method

1: $\mathbf{r}_0 = \mathbf{a}_N(\omega_0)$
2: $\mathbf{p}_0 = \mathbf{r}_0$
3: $\mathbf{w}_0 = \mathbf{0}$
4: **for** $i = 0$ to $N-1$ **do**
5: $\quad \alpha_i = (\mathbf{r}_i^H \mathbf{r}_i)/(\mathbf{p}_i^H \widehat{\mathbf{R}}_{\mathbf{xx}} \mathbf{p}_i)$
6: $\quad \mathbf{w}_{i+1} = \mathbf{w}_i + \alpha_i \mathbf{p}_i$
7: $\quad \mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i \widehat{\mathbf{R}}_{\mathbf{xx}} \mathbf{p}_i$
8: $\quad$ **if** $\|\mathbf{r}_{i+1}\|^2 < \epsilon$ **then**
9: $\quad\quad$ **break** ▷ End iteration if residual approaches zero
10: $\quad$ **end if**
11: $\quad \beta_i = (\mathbf{r}_{i+1}^H \mathbf{r}_{i+1})/(\mathbf{r}_i^H \mathbf{r}_i)$
12: $\quad \mathbf{p}_{i+1} = \mathbf{r}_{i+1} + \beta_i \mathbf{p}_i$
13: **end for**
14: **return** $\mathbf{w}_{i+1}$

solving a set of positive definite linear equations [38]. Hence, (23) can be solved by the CG method, which is summarized as a pseudocode in Algorithm 1 shown in the table. In the CG method, we construct a set of vectors $\mathbf{p}_0, \ldots, \mathbf{p}_{N-1}$ that form a basis for $\mathbb{C}^N$ and are mutually conjugate with respect to $\widehat{\mathbf{R}}_{\mathbf{xx}}$, i.e., $\mathbf{p}_i^H \widehat{\mathbf{R}}_{\mathbf{xx}} \mathbf{p}_k = 0$ for all $i \neq k$. Then the final solution $\mathbf{w}$ can be expressed as

$$\mathbf{w} = \sum_{i=0}^{N-1} \alpha_i \mathbf{p}_i \tag{24}$$

for some α_i. The basis vectors $\mathbf{p}_i$ and coefficients α_i are obtained iteratively as in Algorithm 1. The quantity $\mathbf{w}_{i+1} = \sum_{k=0}^{i} \alpha_k \mathbf{p}_k$ is the partial solution in iteration $i+1$, and $\mathbf{r}_{i+1} = \mathbf{a}_N(\omega_0) - \widehat{\mathbf{R}}_{\mathbf{xx}} \mathbf{w}_{i+1}$ is the corresponding residual. Line 9 serves as a checkpoint for ending the CG iterations. If the residual $\mathbf{r}_{i+1} = \mathbf{0}$, the CG iterations should be terminated. In practice, we check if $\|\mathbf{r}_{i+1}\|^2 < \epsilon$ for some small positive ϵ. For more details of the CG method, one may refer to [34]. The key point of using the CG method here is that $\widehat{\mathbf{R}}_{\mathbf{xx}}$ appears only in the linear operation $\mathbf{q}_i = \widehat{\mathbf{R}}_{\mathbf{xx}} \mathbf{p}_i$ in Lines 5 and 7. This *can be computed using AC* in a way similar to (11).

The proposed distributed Capon beamforming based on CG method is presented in Algorithm 2 shown in the table. It is essentially using the CG method to solve (23) and making the algorithm distributed. Some new variables are defined in Algorithm 2 so that some partial results can be reused. Throughout the algorithm, the vectors $\mathbf{p}_{i,p}, \mathbf{q}_{i,p}, \mathbf{r}_{i,p}, \mathbf{w}_{i,p} \in \mathbb{C}^Q$ are subvectors of $\mathbf{p}_i, \mathbf{q}_i, \mathbf{r}_i, \mathbf{w}_i \in \mathbb{C}^N$, respectively, corresponding to node p, and are accessible only to node p. Moreover, the computations in Lines 10, 15, 16, and 24 are done locally at each node *in parallel*. AC appears in the algorithm in three places, Lines 7, 12, and 18. The inner products in the AC arguments are also computed locally at node p in parallel. Line 7 is the dominant operation for communication, leading to $O(KNI_{\text{ac}})$ communication cost per edge, where I_{ac} is the number of AC iterations. After computing the Capon beamformer (21) using Algorithm 2, we simply have to apply the beamformer as in (18). This can be also done using AC since

$$z_{\text{BF}} = P \cdot \text{AC}_p(\mathbf{h}_p^H \mathbf{x}_p). \tag{25}$$

Here $\mathbf{h}_p \in \mathbb{C}^Q$ is the subvector of $\mathbf{h}$ corresponding to node p.

Algorithm 2 Distributed design of Capon beamformer

1: $\mathbf{r}_0 = \mathbf{a}_N(\omega_0)$
2: $a_0 = \mathbf{r}_0^H \mathbf{r}_0$
3: $\mathbf{p}_0 = \mathbf{r}_0$
4: $\mathbf{w}_0 = \mathbf{0}$
5: **for** $i = 0$ to $N-1$ **do**
6: $\quad$ **for** $k = 1$ to K **do**
7: $\quad\quad t[k] = P \cdot \text{AC}_p(\mathbf{x}_p^H[k] \mathbf{p}_{i,p})$
8: $\quad$ **end for**
9: $\quad$ **for** $p = 0$ to $P-1$ **do**
10: $\quad\quad \mathbf{q}_{i,p} = \frac{1}{K} \sum_{k=1}^{K} \mathbf{x}_p[k] t[k]$ ▷ Compute $\mathbf{q}_i = \widehat{\mathbf{R}}_{\mathbf{xx}} \mathbf{p}_i$
11: $\quad$ **end for**
12: $\quad b_i = P \cdot \text{AC}_p(\mathbf{p}_{i,p}^H \mathbf{q}_{i,p})$
13: $\quad \alpha_i = a_i / b_i$
14: $\quad$ **for** $p = 0$ to $P-1$ **do**
15: $\quad\quad \mathbf{w}_{i+1,p} = \mathbf{w}_{i,p} + \alpha_i \mathbf{p}_{i,p}$
16: $\quad\quad \mathbf{r}_{i+1,p} = \mathbf{r}_{i,p} - \alpha_i \mathbf{q}_{i,p}$
17: $\quad$ **end for**
18: $\quad a_{i+1} = P \cdot \text{AC}_p(\mathbf{r}_{i+1,p}^H \mathbf{r}_{i+1,p})$
19: $\quad$ **if** $a_{i+1} < \epsilon$ **then**
20: $\quad\quad$ **break** ▷ End iteration if residual approaches zero
21: $\quad$ **end if**
22: $\quad \beta_i = a_{i+1} / a_i$
23: $\quad$ **for** $p = 0$ to $P-1$ **do**
24: $\quad\quad \mathbf{p}_{i+1,p} = \mathbf{r}_{i+1,p} + \beta_i \mathbf{p}_{i,p}$
25: $\quad$ **end for**
26: **end for**
27: **return** $\mathbf{w}_{i+1}$

Our idea of using the CG method to compute the inverse of the covariance matrix can be applied to situations other than Capon beamforming. It can be used for any problem that requires the inversion of an array output covariance. For example, we can derive a distributed algorithm for the method of space-time adaptive processing (STAP) for MIMO radar systems proposed in [39].

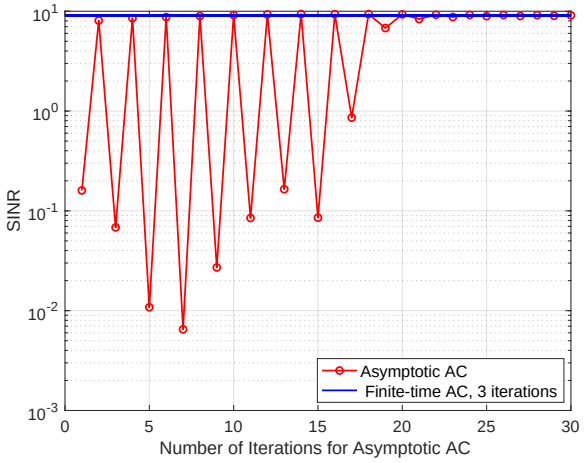
A numerical example is next given to demonstrate the effectiveness of the proposed distributed Capon beamforming. We consider the network as in [13] composed of $P = 6$ nodes, and the neighboring sets are

$$\begin{aligned} &\mathcal{N}_0 = \{1,2\}, \mathcal{N}_1 = \{0,2\}, \mathcal{N}_2 = \{0,1,3\}, \\ &\mathcal{N}_3 = \{2,4,5\}, \mathcal{N}_4 = \{3,5\}, \mathcal{N}_5 = \{3,4\}. \end{aligned} \tag{26}$$

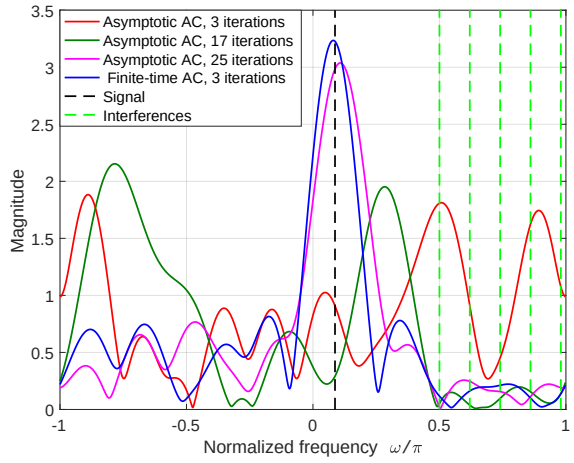
For this network, using the finite-time AC method in [11], we can achieve exact convergence in 3 iterations. This is because, in this case, the Laplacian $\mathbf{L}$ has $R = 4$ distinct values. The weight matrix $\mathbf{W}(t)$ of each iteration can be found using (7) and (8). We compare this finite-time AC method with the asymptotic AC method using the optimal symmetric weight matrix proposed in [10]. Each node is a ULA with $Q = 2$ sensors and they together form a 12-sensor ULA. The signal of interest is at angle $\theta = 5^\circ$, which is assumed known. There are 5 interfering sources with DOAs $\omega = 0.5\pi, 0.62\pi, 0.74\pi, 0.86\pi$, and 0.98π. (Recall $\omega = \pi \sin\theta$.) All sources are uncorrelated with power 1. The noise variance is $\sigma_e^2 = 1$. Output SINRs for various number of iterations for asymptotic AC are shown in Fig. 2(a). It does not make sense to consider the SINRs of finite-time AC before

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing



(a)



(b)

Fig. 2. Distributed Capon beamforming with asymptotic AC and finite-time AC methods. The number of iterations for finite-time AC is fixed at 3. (a) Output SINR. (b) Typical beamformer responses.

it reaches exact convergence with a sufficient finite number of iterations, so we just fix it to be that number, 3 in this example. We use 500 snapshots and average 2000 Monte Carlo runs to get the plot. Indeed, finite-time AC converges much faster than asymptotic AC. Although asymptotic AC gets SINRs similar to finite-time AC also in some of the small numbers of iterations, we cannot know which iterations yield larger SINRs if we do not know the ground truth, so in practice we still have to use a large enough number of iterations for asymptotic AC, like 20 in this example. Hence, we propose to use finite-time AC in our distributed algorithms. Note that the centralized Capon beamformer yields exactly the same SINR as the finite-time AC method, so it is not plotted. Typical beamformer responses are also shown in Fig. 2(b). The beam pattern of asymptotic AC gradually converges to that of finite-time AC as iterations progress.

B. Distributed MUSIC and Root-MUSIC

In algorithms such as MUSIC [2] and root-MUSIC [3], we first compute the EVD of the covariance matrix $\mathbf{R_{xx}}$ of the array output $\mathbf{x}$ and obtain

$$\mathbf{R_{xx}} = \mathbf{E}_{\rm s}\mathbf{\Lambda}_{\rm s}\mathbf{E}_{\rm s}^H + \mathbf{E}_{\rm n}\mathbf{\Lambda}_{\rm n}\mathbf{E}_{\rm n}^H, \tag{27}$$

where $\mathbf{E}_{\rm s} = [\mathbf{e}_1 \cdots \mathbf{e}_D]$ and $\mathbf{E}_{\rm n} = [\mathbf{e}_{D+1} \cdots \mathbf{e}_N]$ contain the signal and noise eigenvectors respectively with $\mathbf{e}_k^H\mathbf{e}_m = \delta[k-m]$, and $\mathbf{\Lambda}_{\rm s}$ and $\mathbf{\Lambda}_{\rm n}$ are diagonal matrices containing the corresponding eigenvalues. In the MUSIC algorithm, we evaluate the MUSIC spectrum

$$P(\omega) = (\mathbf{a}_N^H(\omega)\mathbf{E}_{\rm n}\mathbf{E}_{\rm n}^H\mathbf{a}_N(\omega))^{-1} \tag{28}$$

on a dense grid of potential DOAs and identify local maxima as the DOA estimates. This is because theoretically, if ω is a true DOA, then the steering vector $\mathbf{a}_N(\omega)$ is orthogonal to noise subspace $\mathbf{E}_{\rm n}$, i.e., $\mathbf{a}_N^H(\omega)\mathbf{E}_{\rm n}\mathbf{E}_{\rm n}^H\mathbf{a}_N(\omega) = 0$. The converse (i.e., if $\mathbf{a}_N^H(\omega)\mathbf{E}_{\rm n}\mathbf{E}_{\rm n}^H\mathbf{a}_N(\omega) = 0$, then ω is a true DOA) is also true for a ULA (given the number of sources $D < N$), but not true in general for a non-ULA [40]–[42]. Our focus here is the development of distributed algorithms. We will see that these algorithms are insensitive to whether the array is uniform or not, and we do not focus on well-known identifiability issues here. In practice, $\mathbf{R_{xx}}$ and $\mathbf{e}_i$ are replaced by their finite-snapshot estimates $\widehat{\mathbf{R}}_{\mathbf{xx}}$ and $\hat{\mathbf{e}}_i$ respectively.

Root-MUSIC is a variation of the MUSIC algorithm. It avoids the evaluation of MUSIC spectrum on a dense grid by computing the roots of a polynomial, thereby improving accuracy, and reducing complexity. Thus, instead of finding local maxima of $P(\omega)$, we find roots ρ_i of the polynomial

$$f(z) = \underline{\mathbf{a}}_N^T(z)\hat{\mathbf{E}}_{\rm n}\hat{\mathbf{E}}_{\rm n}^H\overline{\mathbf{a}}_N(z), \tag{29}$$

where $\underline{\mathbf{a}}_N(z) = z^{z_N-1}[1 \;\; z^{-z_1} \cdots z^{-z_{N-1}}]^T$ and $\overline{\mathbf{a}}_N(z) = [1 \;\; z^{z_1} \cdots z^{z_{N-1}}]^T$ (with z_i denoting sensor locations as before). This polynomial is obtained by replacing $e^{j\omega}$ with z in $P^{-1}(\omega)$ and then multiplying it with z^{z_N-1}. The DOAs ω_i are obtained from the arguments of ρ_i.

Now we show how to do MUSIC and root-MUSIC in our distributed setting. In this case, the eigenvector estimates are obtained using the distributed power method in Sec. II, and the results are $\hat{\mathbf{e}}_i = [\hat{\mathbf{e}}_{i,0}^T \cdots \hat{\mathbf{e}}_{i,P-1}^T]^T$ with $\hat{\mathbf{e}}_{i,p}$ only known to node p. Then the MUSIC spectrum can be evaluated as

$$P(\omega) = \left(\sum_{i=D+1}^{N} \left| P \cdot \mathrm{AC}_p(\hat{\mathbf{e}}_{i,p}^H \mathbf{a}_{N,p}(\omega)) \right|^2 \right)^{-1}, \tag{30}$$

where $\mathbf{a}_{N,p}(\omega) \in \mathbb{C}^Q$ is the subvector of $\mathbf{a}_N(\omega)$ corresponding to node p. Alternatively, since $\hat{\mathbf{E}} \triangleq [\hat{\mathbf{E}}_{\rm s} \;\; \hat{\mathbf{E}}_{\rm n}]$ is a unitary matrix so that $\hat{\mathbf{E}}_{\rm s}\hat{\mathbf{E}}_{\rm s}^H + \hat{\mathbf{E}}_{\rm n}\hat{\mathbf{E}}_{\rm n}^H = \mathbf{I}$, we also have

$$P(\omega) = (\mathbf{a}_N^H(\omega)(\mathbf{I} - \hat{\mathbf{E}}_{\rm s}\hat{\mathbf{E}}_{\rm s}^H)\mathbf{a}_N(\omega))^{-1} \tag{31}$$

$$= (N - \mathbf{a}_N^H(\omega)\hat{\mathbf{E}}_{\rm s}\hat{\mathbf{E}}_{\rm s}^H\mathbf{a}_N(\omega))^{-1} \tag{32}$$

$$= \left(N - \sum_{i=1}^{D} \left| P \cdot \mathrm{AC}_p(\hat{\mathbf{e}}_{i,p}^H \mathbf{a}_{N,p}(\omega)) \right|^2 \right)^{-1}. \tag{33}$$

Typically, (33) is advantageous over (30) in terms of computation and communication among nodes because dominant eigenvectors $\hat{\mathbf{e}}_1, \ldots, \hat{\mathbf{e}}_D$ are obtained first by the distributed power method, unless D is almost as large as N. The communication cost per edge for evaluating $P(\omega)$ using (33) is $O(DGI_{\rm ac})$, where G is the number of grid points used for ω.

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

Algorithm 3 Distributed root-MUSIC

Randomly initialize $z_{0,1},\ldots,z_{0,2z_{N-1}}$
for $i=0$ to $I_{\max}$ **do**
  for $k=1$ to $2z_{N-1}$ **do**
    $\overline{\mathbf{u}}=P\cdot\mathrm{AC}_p(\hat{\mathbf{E}}_{\mathrm{s},p}^H\overline{\mathbf{a}}_{N,p}(z_{i,k}))$
    $\underline{\mathbf{u}}=P\cdot\mathrm{AC}_p(\hat{\mathbf{E}}_{\mathrm{s},p}^T\underline{\mathbf{a}}_{N,p}(z_{i,k}))$
    $\overline{\mathbf{v}}=P\cdot\mathrm{AC}_p(\hat{\mathbf{E}}_{\mathrm{s},p}^H\overline{\mathbf{b}}_{N,p}(z_{i,k}))$
    $\underline{\mathbf{v}}=P\cdot\mathrm{AC}_p(\hat{\mathbf{E}}_{\mathrm{s},p}^T\underline{\mathbf{b}}_{N,p}(z_{i,k}))$
    $s=Nz_{i,k}^{z_{N-1}}-\underline{\mathbf{u}}^T\overline{\mathbf{u}}$
    $t=Nz_{N-1}z_{i,k}^{z_{N-1}-1}-\underline{\mathbf{v}}^T\overline{\mathbf{u}}-\underline{\mathbf{u}}^T\overline{\mathbf{v}}$
    $z_{i+1,k}=z_{i,k}-[t/s-\sum_{l\neq k}(1/(z_{i,k}-z_{i,l}))]^{-1}$
  end for
  if $\sum_k|z_{i+1,k}-z_{i,k}|^2<\epsilon$ **then**
    break ▷ End iteration if all roots converge
  end if
end for
return $z_{i+1,1},\ldots,z_{i+1,2z_{N-1}}$

For distributed root-MUSIC, we can similarly consider either (29) or

$$f(z)=Nz^{z_{N-1}}-\underline{\mathbf{a}}_N^T(z)\hat{\mathbf{E}}_{\mathrm{s}}\hat{\mathbf{E}}_{\mathrm{s}}^H\overline{\mathbf{a}}_N(z). \tag{34}$$

In the following, we use the form in (34) to derive our distributed algorithm because it is easier to obtain $\hat{\mathbf{E}}_{\mathrm{s}}$ than $\hat{\mathbf{E}}_{\mathrm{n}}$. A similar algorithm can be derived for (29). To understand the crucial step that is required here, consider $\hat{\mathbf{E}}_{\mathrm{s}}=[\hat{\mathbf{E}}_{\mathrm{s},0}^T\cdots\hat{\mathbf{E}}_{\mathrm{s},P-1}^T]^T$, where $\hat{\mathbf{E}}_{\mathrm{s},p}\in\mathbb{C}^{Q\times D}$ contains the local subvectors of eigenvectors for node p. To explicitly compute all entries of $\hat{\mathbf{E}}_{\mathrm{s}}\hat{\mathbf{E}}_{\mathrm{s}}^H$, we need to compute $\hat{\mathbf{E}}_{\mathrm{s},p}\hat{\mathbf{E}}_{\mathrm{s},q}^H$ for each pair of p and q. This cannot be done without a fusion center or sharing data among all nodes. Instead of explicitly computing $\hat{\mathbf{E}}_{\mathrm{s}}\hat{\mathbf{E}}_{\mathrm{s}}^H$, we propose to use the *Aberth method* [35] as our polynomial-rooting algorithm and show that *the method can be done in a distributed way* based on AC. Given a polynomial $f(z)$ of degree n, the Aberth method is an algorithm that finds all roots simultaneously by the iteration

$$z_{i+1,k}=z_{i,k}-\left[\frac{f'(z_{i,k})}{f(z_{i,k})}-\sum_{\substack{l=1\\l\neq k}}^{n}\frac{1}{z_{i,k}-z_{i,l}}\right]^{-1} \tag{35}$$

for $k=1,\ldots,n$. Here $z_{i,k}$ stands for the kth root in the ith iteration, and $f'(z_{i,k})$ is the first derivative of $f(z)$ evaluated at $z=z_{i,k}$. The key point of using the Aberth method is that $\hat{\mathbf{E}}_{\mathrm{s}}$ appears only in $f'(z_{i,k})$ and $f(z_{i,k})$, and we can show that distributed computation of these can be done using AC. After plugging (34) into (35) and some derivations, we can realize distributed root-MUSIC as summarized in Algorithm 3 shown in the table. In this algorithm, $\overline{\mathbf{a}}_{N,p}(z),\underline{\mathbf{a}}_{N,p}(z),\overline{\mathbf{b}}_{N,p}(z),\underline{\mathbf{b}}_{N,p}(z)\in\mathbb{C}^Q$ are subvectors of $\overline{\mathbf{a}}_N(z),\underline{\mathbf{a}}_N(z),\frac{\mathrm{d}}{\mathrm{d}z}\overline{\mathbf{a}}_N(z)$, and $\frac{\mathrm{d}}{\mathrm{d}z}\underline{\mathbf{a}}_N(z)$, respectively, corresponding to node p. The vectors $\overline{\mathbf{a}}_{N,p}(z_{i,k})$, $\underline{\mathbf{a}}_{N,p}(z_{i,k})$, $\overline{\mathbf{b}}_{N,p}(z_{i,k})$, $\underline{\mathbf{b}}_{N,p}(z_{i,k})$ can be evaluated at node p because each $z_{i,k}$ is known to all nodes. The matrix multiplications in the four AC arguments are computed locally at node p in parallel. The iterative algorithm stops either when all roots converge, i.e.,

$$\sum_k|z_{i+1,k}-z_{i,k}|^2<\epsilon \tag{36}$$

for some small positive ϵ, or when a predefined maximum number of iterations $I_{\max}$ is reached. The total communication cost per edge is $O(z_{N-1}DI_{\mathrm{ac}}I_{\mathrm{ab}})$, where I_{ab} is the number of Aberth iterations. A numerical example is given in Sec. IV-D.

The Aberth method works even for non-distinct roots [35]. One scenario that hinders convergence is when the roots are symmetrically positioned in the complex plane with respect to some line, and the initial guess of the roots makes them also symmetrically placed with respect to the line. However, this happens with probability zero in our case due to two reasons. First, the actual roots resulting from a finite number of snapshots of the array output are not symmetrical with probability one. Second, we initialize the roots randomly with some continuous probability distribution, so they are not symmetrical with probability one. Hence, essentially it will always converge in our case.

C. Distributed ESPRIT

ESPRIT [4] is another commonly used subspace-based DOA estimation algorithm. For ESPRIT, we assume that each node is a Q-sensor ULA with the same sensor spacing $\lambda/2$, but the displacements between the subarrays are unknown, so the entire array can be non-uniform. This array setting follows [13]. In [13], ESPRIT based on least-squares estimates (LS-ESPRIT) was shown to be realizable in this distributed setting. It is well-known that total least-squares (TLS) ESPRIT produces more accurate estimates than LS-ESPRIT [43]. In this subsection, we will show that distributed TLS-ESPRIT can also be done, and we will present distributed LS-ESPRIT and TLS-ESPRIT in a unified framework.

In ESPRIT, we require two groups of sensors with a shift invariance between them. Here we define the first group to be the first $Q-1$ sensors of each node and the second group to be the last $Q-1$ sensors of each node. Hence, the shift invariance is the unit sensor spacing $\lambda/2$. Equivalently, consider the selection matrices

$$\bar{\mathbf{J}}_1=[\mathbf{I}_{Q-1}\ \mathbf{0}_{Q-1}] \tag{37}$$

and

$$\bar{\mathbf{J}}_2=[\mathbf{0}_{Q-1}\ \mathbf{I}_{Q-1}], \tag{38}$$

where $\mathbf{0}_{Q-1}$ is the $(Q-1)\times 1$ zero vector. Moreover, let

$$\mathbf{J}_l=\mathbf{I}_P\otimes\bar{\mathbf{J}}_l \tag{39}$$

for $l=1,2$, where $\otimes$ denotes the Kronecker product. Then in ESPRIT, we have to select the rows of signal eigenvectors $\hat{\mathbf{E}}_{\mathrm{s}}$ corresponding to each group, i.e., $\hat{\mathbf{E}}_l=\mathbf{J}_l\hat{\mathbf{E}}_{\mathrm{s}}$ for $l=1,2$. Ideally, $\hat{\mathbf{E}}_1$ and $\hat{\mathbf{E}}_2$ have the same column space if we have infinite snapshots. In practice, we find the LS or TLS solution [4] to $\hat{\mathbf{E}}_1\mathbf{\Psi}=\hat{\mathbf{E}}_2$. The LS solution is

$$\mathbf{\Psi}_{\mathrm{LS}}=(\hat{\mathbf{E}}_1^H\hat{\mathbf{E}}_1)^{-1}(\hat{\mathbf{E}}_1^H\hat{\mathbf{E}}_2). \tag{40}$$

Then the DOA estimates of LS-ESPRIT are [4]

$$\hat{\omega}_{\mathrm{LS},k}=\arg(\lambda_k(\mathbf{\Psi}_{\mathrm{LS}})), \tag{41}$$

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

the argument of the kth eigenvalue of $\mathbf{\Psi}_{\rm LS}$. To compute the TLS solution, we consider

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{1,1} & \mathbf{F}_{1,2} \\ \mathbf{F}_{2,1} & \mathbf{F}_{2,2} \end{bmatrix}, \tag{42}$$

where $\mathbf{F}_{l,m} = \hat{\mathbf{E}}_l^H \hat{\mathbf{E}}_m$ for $l = 1, 2$ and $m = 1, 2$, and compute its EVD

$$\mathbf{F} = \bar{\mathbf{E}} \bar{\Lambda} \bar{\mathbf{E}}^H, \tag{43}$$

with the eigenvalues in descending order. Then we decompose the matrix $\bar{\mathbf{E}} \in \mathbb{C}^{2D \times 2D}$ into $D \times D$ submatrices, i.e.,

$$\bar{\mathbf{E}} = \begin{bmatrix} \bar{\mathbf{E}}_{1,1} & \bar{\mathbf{E}}_{1,2} \\ \bar{\mathbf{E}}_{2,1} & \bar{\mathbf{E}}_{2,2} \end{bmatrix}. \tag{44}$$

The TLS solution is then

$$\mathbf{\Psi}_{\rm TLS} = -\bar{\mathbf{E}}_{1,2} \bar{\mathbf{E}}_{2,2}^{-1}, \tag{45}$$

and the DOA estimates of TLS-ESPRIT are

$$\hat{\omega}_{{\rm TLS},k} = \arg(\lambda_k(\mathbf{\Psi}_{\rm TLS})). \tag{46}$$

Now we show how to do ESPRIT in our distributed setting. In [13], the authors showed how to do this for LS-ESPRIT. We now give a unified derivation for distributed ESPRIT, which works whether it is LS- or TLS-ESPRIT. For $l = 1, 2$ and $m = 1, 2$, we have

$$[\hat{\mathbf{E}}_l^H \hat{\mathbf{E}}_m]_{i,k} = \sum_{p=0}^{P-1} (\bar{\mathbf{J}}_l \hat{\mathbf{e}}_{i,p})^H (\bar{\mathbf{J}}_m \hat{\mathbf{e}}_{k,p}) \tag{47}$$

$$= P \cdot {\rm AC}_p((\bar{\mathbf{J}}_l \hat{\mathbf{e}}_{i,p})^H (\bar{\mathbf{J}}_m \hat{\mathbf{e}}_{k,p})) \tag{48}$$

for each (i, k)-entry. Again, $\hat{\mathbf{e}}_i = [\hat{\mathbf{e}}_{i,0}^T \cdots \hat{\mathbf{e}}_{i,P-1}^T]^T$ is the ith eigenvector with $\hat{\mathbf{e}}_{i,p} \in \mathbb{C}^Q$ being the subvector corresponding to node p. Thus, $(\bar{\mathbf{J}}_l \hat{\mathbf{e}}_{i,p})^H (\bar{\mathbf{J}}_m \hat{\mathbf{e}}_{k,p})$ can be computed locally at node p. Then with AC, each node can obtain the entire matrix $\mathbf{F}$. Hence for LS-ESPRIT, we can compute

$$\mathbf{\Psi}_{\rm LS} = \mathbf{F}_{1,1}^{-1} \mathbf{F}_{1,2} \tag{49}$$

and then (41) locally at each node. Likewise, for TLS-ESPRIT, (43) through (46) can also be realized locally at each node. This leads to an increase in the total computational complexity since all the nodes are performing the same operations. An alternative would be for one node to do the computations and then send the final DOA estimates to the other nodes. This is a tradeoff between computation and communication. The communication cost per edge for both LS-ESPRIT and TLS-ESPRIT is $O(D^2 I_{\rm ac})$, though TLS-ESPRIT has twice communication cost of LS-ESPRIT because TLS-ESPRIT requires the entire matrix $\mathbf{F}$ while LS-ESPRIT requires only $\mathbf{F}_{1,1}$ and $\mathbf{F}_{1,2}$. This is a tradeoff for getting better DOA estimates with TLS-ESPRIT.

A numerical example is given next to demonstrate the effectiveness of the proposed distributed TLS-ESPRIT. We consider the same network (26) and use the finite-time AC method with 3 iterations as explained in Sec. III-A. We compare distributed TLS-ESPRIT with distributed LS-ESPRIT [13] and centralized TLS-ESPRIT [4]. Here the distributed power method shown in Sec. II is used to estimate eigenvectors of the array output covariance for distributed TLS-ESPRIT and distributed LS-ESPRIT, whereas ideal EVD of the covariance is assumed for centralized TLS-ESPRIT. The sensor locations of the pth node are $15p, 15p + 1, \ldots, 15p + 9$, so each node is a ULA with $Q = 10$ sensors, and there are 60 sensors in total. Note that the entire array is not a ULA. The displacement 15 between the subarrays is arbitrary and assumed unknown. There are 6 DOAs at angles $\theta = -5°, -3° - 1°, \ldots, 5°$. The sources are uncorrelated with power -5 dB. The noise variance is $\sigma_e^2 = 1$. We assume the number of DOAs is known. The number of power iterations is $I_{\rm pm} = 5$. Root mean square errors (RMSEs) of DOA estimates for various number of snapshots are shown in Fig. 3. In this paper, whenever we mention RMSEs in detected source angles, we refer to averaging square errors measured in ω over all involved DOAs. We average 5000 Monte Carlo runs to get the plot. As expected, the RMSE of distributed TLS-ESPRIT is smaller than that of distributed LS-ESPRIT. The RMSE of centralized TLS-ESPRIT is even smaller at 40 snapshots because ideal EVD is used for it. However for 50 or more snapshots, distributed TLS-ESPRIT using only 5 power iterations performs almost the same as centralized TLS-ESPRIT.

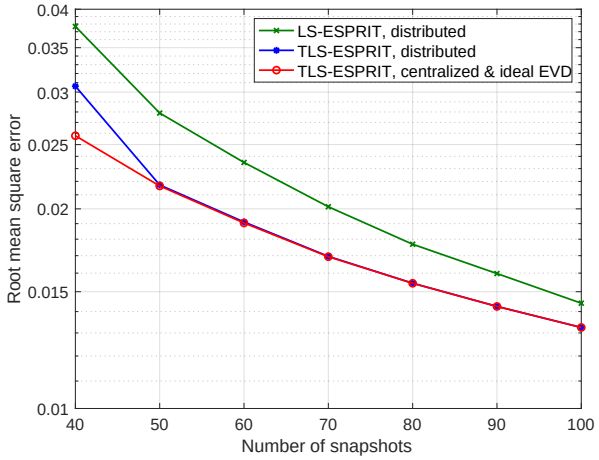


Fig. 3. RMSE of DOA estimates using distributed LS-ESPRIT, distributed TLS-ESPRIT, and centralized TLS-ESPRIT.

D. Distributed FOCUSS

The FOCUSS algorithm [6], [7] for finding sparse solutions to an underdetermined system can also be used to estimate DOAs. We first briefly review the centralized FOCUSS for the case of multiple measurement vectors [7], or multiple snapshots in our language. To use FOCUSS for DOA estimation, a dictionary $\mathbf{D}$ of steering vectors $\mathbf{a}_N(\bar{\omega}_i)$ on a dense grid of potential DOAs $\{\bar{\omega}_i\}_{i=1}^d$ is considered, and the goal is to find sparse signals $\mathbf{Q} = [\mathbf{q}(1)\ \mathbf{q}(2) \cdots \mathbf{q}(K)]^T$ that well represent the K snapshots of array output

$$\mathbf{X} = \underbrace{[\mathbf{a}_N(\bar{\omega}_1)\ \mathbf{a}_N(\bar{\omega}_2)\ \cdots\ \mathbf{a}_N(\bar{\omega}_d)]}_{\text{dictionary } \mathbf{D}} \mathbf{Q} + \mathbf{E}. \tag{50}$$

Here $\mathbf{X} = [\mathbf{x}(1)\ \mathbf{x}(2) \cdots \mathbf{x}(K)]$ and $\mathbf{E} = [\mathbf{e}(1)\ \mathbf{e}(2) \cdots \mathbf{e}(K)]$ are the K snapshots of array outputs and noise, respectively. The number of grid points d is typically much larger than D, the number of sources. If D is not too large [7] (the exact

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

TABLE I
COMMUNICATION COSTS OF DISTRIBUTED AND CENTRALIZED DOA ESTIMATION METHODS

Algorithm	Communication cost	Typical numbers
Distributed MUSIC	$O(DKI_{\rm ac}I_{\rm pm} + DGI_{\rm ac})$	3000
Distributed root-MUSIC	$O(DKI_{\rm ac}I_{\rm pm} + z_{N-1}DI_{\rm ac}I_{\rm ab})$	8940
Distributed TLS-ESPRIT	$O(DKI_{\rm ac}I_{\rm pm} + D^2I_{\rm ac})$	1812
Distributed FOCUSS	$O(dKI_{\rm ac}I_{\rm fo})$	9×10^5
All centralized methods	$O(KN)$	12000

bound depending on N and K), there exists a solution to (50) such that $\mathbf{q}(k)$ has a common sparse pattern across all snapshots. The locations of nonzero entries of $\mathbf{q}(k)$ reveal the DOAs, and the values represent the source amplitudes. FOCUSS is an iterative algorithm for solving such a problem. It is initialized with

$$\mathbf{Q}_0 = \mathbf{D}^+\mathbf{X}. \tag{51}$$

Then we iterate

$$\mathbf{W}_{n+1} = \mathrm{diag}(\|\mathbf{q}_{n,1}\|^{1-p/2}, \ldots, \|\mathbf{q}_{n,d}\|^{1-p/2}) \tag{52}$$

$$\mathbf{Q}_{n+1} = \mathbf{W}_{n+1}(\mathbf{D}\mathbf{W}_{n+1})^+\mathbf{X} \tag{53}$$

for $n = 0, 1, \ldots,$ until convergence. Here $\mathbf{q}_{n,i}^T$ is the ith row of $\mathbf{Q}_n$, and $p \in [0, 1]$ is a parameter to be chosen so that the solution aims to minimize the ℓ_p diversity measure [7], i.e., the ℓ_p pseudo-norm of the vector whose elements are the ℓ_2 norms of the rows of $\mathbf{Q}$.

Now we show how to do FOCUSS in our distributed setting, where

$$\mathbf{X} = [\mathbf{X}_0^T\ \mathbf{X}_1^T \cdots \mathbf{X}_{P-1}^T]^T \tag{54}$$

with $\mathbf{X}_p \in \mathbb{C}^{Q\times K}$ stored locally at node p. Let $\mathbf{B}_0 = \mathbf{D}^+ = [\mathbf{B}_{0,0} \cdots \mathbf{B}_{0,P-1}]$ with each $\mathbf{B}_{0,p} \in \mathbb{C}^{d\times Q}$. Since the dictionary $\mathbf{D}$ is known to all nodes, $\mathbf{B}_{0,p}$ can also be obtained at node p. Hence, (51) can be computed from

$$\mathbf{Q}_0 = P \cdot \mathrm{AC}_p(\mathbf{B}_{0,p}\mathbf{X}_p). \tag{55}$$

Let $\mathbf{B}_{n+1} = \mathbf{W}_{n+1}(\mathbf{D}\mathbf{W}_{n+1})^+ = [\mathbf{B}_{n+1,0} \cdots \mathbf{B}_{n+1,P-1}]$ with each $\mathbf{B}_{n+1,p} \in \mathbb{C}^{d\times Q}$. Then (53) can be computed from

$$\mathbf{Q}_{n+1} = P \cdot \mathrm{AC}_p(\mathbf{B}_{n+1,p}\mathbf{X}_p). \tag{56}$$

Since each $\mathbf{Q}_{n+1}$ is the result of AC, it is known to all nodes, so (52) can also be formed at each node. Thus, each $\mathbf{B}_{n+1,p}$ can indeed be obtained at node p. The fact that only linear operations on array outputs are involved in the iterations makes FOCUSS readily realizable using AC. As in distributed ESPRIT, computing $(\mathbf{D}\mathbf{W}_{n+1})^+$ at all the nodes leads to an increase in the total computational complexity since they are performing the same operations. An alternative would be for one node to do the computations and then send the results to the other nodes. This is a tradeoff between computation and communication. The communication cost per edge is $O(dKI_{\rm ac}I_{\rm fo})$, where $I_{\rm fo}$ is the number of FOCUSS iterations. A numerical example is given in Sec. IV-D.

Distributed FOCUSS evidently has broader applications than DOA estimation. It can be used for any problem that requires a sparse solution to an underdetermined system [7].

E. Communication, Computation, and Storage

We now compare the communication costs of the proposed distributed algorithms and centralized algorithms. Using the results in Sec. II-B for distributed power method and in this section for the DOA estimation methods, the communication costs per edge for the various distributed algorithms are summarized in Table I. For a ULA, $z_{N-1} = N-1$. Typically, the number of DOAs D is much smaller than either of the number of grid points G used for ω in MUSIC, number of sensors N, number of snapshots K, and number of grid points d used for ω in FOCUSS. How we choose the numbers of iterations $I_{\rm pm}$, $I_{\rm ab}$, and $I_{\rm fo}$ depends on the particular DOA problem settings, but empirically we need a relatively small $I_{\rm pm}$. Also, according to Sec. II-A, $I_{\rm ac} < P$, and the exact number of $I_{\rm ac}$ depends on the network structure. Hence, the communication cost of TLS-ESPRIT is typically the smallest. That is, TLS-ESPRIT is the DOA estimation method particularly suitable for distributed implementations. On the other hand, to implement any centralized DOA estimation method, each node always sends the raw data to a fusion center to do centralized computation. Thus, the communication cost of any centralized DOA estimation method is the same. Given K snapshots of the array output $\mathbf{x}$, the total communication cost across the sensor network is $O(KNT)$, where T is the average distance between each node and the fusion center. The distance between two nodes is the number of edges in a shortest path connecting them. In other words, we have assumed that an efficient routing protocol is used so that each node can send its raw data to the fusion center via a shortest path. In this case, the average communication cost per edge for any centralized algorithm is $O(KNT/|\mathcal{E}|)$. Since we assume the network is a connected graph, $O(1) \le T \le O(P)$ and $O(P) \le |\mathcal{E}| \le O(P^2)$. The possible range of the average communication cost per edge is thus $O(KNP^r)$, where $-2 \le r \le 0$. Hence, whether the communication cost of a distributed algorithm is smaller than the average communication cost of a centralized one depends on the values of the parameters and on the connectivity of the network. Distributed algorithms are typically more advantageous given larger arrays or more snapshots, especially for distributed TLS-ESPRIT.

Besides the average communication cost per edge, another important metric is the maximum communication cost among all the edges because it can determine the existence of a communication bottleneck. Unless the network is almost fully connected, the maximum communication cost for any centralized algorithm is typically close to $O(KN)$, due to an edge nearby the fusion center. By contrast, the maximum communication costs of the distributed algorithms follow

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

the same expressions in Table I because each edge has the same communication cost. Therefore, distributed algorithms are more likely to be better than centralized ones in terms of maximum communication costs, which can determine a communication bottleneck. For comparison, the maximum communication cost of centralized methods is also listed in Table I. Also, we give an example of typical numbers by setting $P = 6, Q = 20, D = 2, K = 100, d = G = 200, I_{\rm ac} = I_{\rm pm} = 3, I_{\rm ab} = 10, I_{\rm fo} = 15$, and $z_{N-1} = 119$. The numbers of iterations are set according to empirical results. We see that distributed TLS-ESPRIT has the smallest communication cost. All the distributed algorithms have smaller costs than centralized ones, except distributed FOCUSS. The large communication cost of FOCUSS is mainly due to the factor dK in the expression. Unless there is some other prior information (e.g., DOAs known to be in some range so that a smaller d can be used), distributed FOCUSS would not be advantageous in terms of communication cost. The numbers in Table I are typical when we assume the number of sources $D \ll N$. The small-D assumption is also made in the literature [14] to show the advantage in communication costs for distributed methods. However, even when this assumption does not hold so that distributed methods do not lower communication costs, other issues can still prevent the use of centralized methods in the first place, as explained in the following.

In the proposed distributed algorithms, the local computation cost at a node depends on the amount of local array data and the number of its neighbors, but not directly on the size of the network. Take TLS-ESPRIT as an example. The computational complexity of (48) locally at node i is $O(D^2Q + D^2 I_{\rm ac}|\mathcal{N}_i|)$, where $|\mathcal{N}_i|$ is the number of its neighbors. Besides, each node only has to store its local array data and AC weights for each edge between each neighbor and itself. It is unlike centralized systems where the fusion center has to store the array data collected from all the nodes to do centralized computation. Therefore, the proposed distributed systems should scale better than centralized systems with fusion centers, as the sensor network expands. The requirement of local storage space and computation power that grow with the network size makes it very challenging to implement centralized algorithms for large networks.

In summary, we have the following key observations.

1) TLS-ESPRIT typically has the smallest communication cost among the proposed distributed algorithms.
2) There are two ways to measure communication cost: average communication cost per edge and maximum communication cost among all the edges. The maximum communication cost can determine a communication bottleneck, so it is more important for the system to be scalable. The two measures do not make a difference for distributed algorithms, but centralized algorithms have larger maximum communication cost than average communication cost. Hence, distributed algorithms are more likely to be better than centralized ones in terms of maximum communication costs.
3) In the proposed distributed algorithms, the local computation and storage at a node depend on the amount of local array data and the number of its neighbors, but not directly on the network size. Thus, these distributed systems should scale better than centralized systems.

IV. Distributed Convolutional Beamspace

In this section, we propose distributed algorithms for a recently introduced beamspace method called *convolutional beamspace (CBS)* [31], [32], and for its variant *Capon-CBS* [44]. Given an N-sensor array with output $\mathbf{x} \in \mathbb{C}^N$, the idea of beamspace is to compute a transformation $\mathbf{y} = \mathbf{T}\mathbf{x} \in \mathbb{C}^B$, where $B < N$, and estimate DOA using $\mathbf{y}$. (For clarity, DOA estimation using $\mathbf{x}$ directly is called *element-space* method.) With a properly designed $\mathbf{T}$, the DOAs falling outside a chosen subband are attenuated, and we focus on estimating only the DOAs in the subband. A bank of transformations $\{\mathbf{T}_i\}$ can be operated in parallel to cover all subbands. One major advantage of beamspace processing is the lowered complexity due to dimensionality reduction ($B < N$). Beamspace methods also tend to have smaller SNR threshold for resolution of closely spaced sources [5], [25], [26] and smaller bias [27], [31].

Classical beamspace methods compromise the Vandermonde structure in the output of a uniform linear array (ULA), so elaborate steps have to be taken to apply root-MUSIC [27] or ESPRIT [28]. By contrast, CBS methods allow root-MUSIC and ESPRIT to be performed directly for ULAs without additional preparation since the Vandermonde structure is preserved. This is achieved by convolving the ULA output with an FIR filter $H(z)$. In traditional CBS [31], the filter $H(z)$ was predefined as a standard lowpass filter, such as the Parks-McClellan filter [45]. But the filter was not designed by taking input statistics into account. In Capon-CBS [44], the filter is designed based on data using the idea of Capon beamforming, so it should do a better job of suppressing the sources falling in the stopband, as they are treated as interference in the Capon method.

The CBS methods have two stages. Stage 1 is the convolutional beamspace (CBS) stage, where the ULA output is convolved with an FIR filter $H(z)$. Stage 2 is the DOA estimation stage. The filter in Stage 1 needs to have a flat passband, and it is a standard lowpass filter in traditional CBS or the *robust Capon beamspace filter* in Capon-CBS. The robust Capon beamspace filter is based on the robust version [46] of Capon beamformer, as proposed in [44] to get a flat passband. Distributed implementation of the CBS filter, i.e., distributed convolution, is introduced in Sec. IV-A. Distributed design of the robust Capon beamspace filter is presented in Sec. IV-B. The DOA estimation algorithms proposed in Sec. III can be used in Stage 2 to form an overall distributed system together with the Stage 1 methods in Secs. IV-A and IV-B. This distributed DOA estimation in Stage 2 is presented in IV-C. Numerical examples are given in Sec. IV-D.

A. Distributed Implementation of the CBS Filter

Convolutional beamspace (CBS) [31], [32] methods are used for ULAs. That is, we assume the sensor locations $z_i = i$ in (2). We convolve the N-sensor ULA output sequence $x(n), 0 \le n \le N-1$ with an FIR filter

$$H(z) = \sum_{n=0}^{L-1} h(n) z^{-n} \tag{57}$$

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

with length $L < N$ and retain *steady state* output samples:

$$\begin{aligned}\mathbf{y} &\triangleq [y(L-1)\ y(L)\cdots y(N-1)]^T \\ &= \mathbf{H}[x(0)\ x(1)\cdots x(N-1)]^T = \mathbf{H}\mathbf{x}, \end{aligned} \tag{58}$$

where $\mathbf{H}$ is a $(N-L+1)\times N$ banded Toeplitz matrix with first row $[h(L-1)\ h(L-2)\cdots h(0)\ 0\ \cdots 0]$ and first column $[h(L-1)\ 0\ \cdots 0]^T$. One can show that [31]

$$\mathbf{y} = \mathbf{A}_L\mathbf{d} + \mathbf{H}\mathbf{e}, \tag{59}$$

where $\mathbf{A}_L$ is a Vandermonde matrix obtained from $\mathbf{A}$ (defined in (1), (2)), by keeping the first $N-L+1$ rows, and $\mathbf{d}$ has elements $d_k = c_k e^{j(L-1)\omega_k} H(e^{j\omega_k})$. The arriving signals with DOAs ω_k are thus filtered by the response $H(e^{j\omega})$. The filter is designed as a standard lowpass filter such as the Parks-McClellan filter [45] in traditional CBS [31], and as a Capon beamspace filter in Capon-CBS [44]. The design of the Capon beamspace filter will be discussed in Sec. IV-B. Assuming signals in the stopband are well attenuated so that $\mathbf{y}$ contains only those DOAs that fall in the passband of $H(e^{j\omega})$, we have $\mathbf{y} \approx \mathbf{A}_{L,0}\bar{\mathbf{d}}_0 + \mathbf{H}\mathbf{e}$. Here $\mathbf{A}_{L,0}$ has D_0 columns of $\mathbf{A}_L$ corresponding to the D_0 sources that fall in the passband, and $\bar{\mathbf{d}}_0$ has the corresponding D_0 rows of $\mathbf{d}$.

To achieve complexity reduction in CBS, we *decimate* $y(n)$ with a uniform downsampler [31]. If the passband of $H(z)$ has width $\approx 2\pi/M$, we can decimate $y(n)$ by the integer M. Let $v(n) = y(n+L-1)$ so that $\mathbf{y} = [v(0)\ v(1)\ \cdots\ v(N-L)]^T$. Then we take the *polyphase components* of $v(n)$ [47]. That is, we define

$$\mathbf{v}_l = [v(l)\ v(l+M)\ \cdots\ v(l+(J-1)M)\ ]^T, \tag{60}$$

for $l = 0,\ldots,M-1$, where $J = \lfloor (N-L+1)/M \rfloor$. It can then be verified that [31]

$$\mathbf{v}_l = \mathbf{A}_{\rm dec}\mathbf{d}_l + \mathbf{D}_l\mathbf{H}\mathbf{e}, \tag{61}$$

where

$$\mathbf{A}_{\rm dec} = [\mathbf{a}_J(M\omega_1)\ \mathbf{a}_J(M\omega_2)\ \cdots\ \mathbf{a}_J(M\omega_D)], \tag{62}$$

$\mathbf{d}_l$ has elements $d_{l,k} = c_k e^{j(L-1+l)\omega_k} H(e^{j\omega_k})$ and $\mathbf{D}_l = [\boldsymbol{\delta}_l\ \boldsymbol{\delta}_{l+M}\ \cdots\ \boldsymbol{\delta}_{l+(J-1)M}]^T$ is a decimation matrix, with $\boldsymbol{\delta}_l$ being the lth standard basis vector for the $(N-L+1)$-dimensional space. The noise term $\mathbf{D}_l\mathbf{H}\mathbf{e}$ can be whitened by making $\mathbf{D}_l\mathbf{H}\mathbf{H}^H\mathbf{D}_l^H = \mathbf{I}$. This is achieved by choosing $H(z)$ as a spectral factor of a Nyquist(M) filter [31]. Since $\mathbf{v}_l$ is represented in terms of the Vandermonde matrix $\mathbf{A}_{\rm dec}$ just like the original ULA output $\mathbf{x}$, standard DOA estimation methods such as root-MUSIC or ESPRIT can be directly applied [31]. Details will be shown in Sec. IV-C. The columns of $\mathbf{A}_{\rm dec}$ are $\mathbf{a}_J(M\omega_k)$ rather than $\mathbf{a}_J(\omega_k)$, so ω_k can be determined only up to $M\omega_k$ mod 2π, creating *ambiguity*. But since ω_k are known to be in the passband of $H(e^{j\omega})$ which has width $2\pi/M$, the ambiguity can be resolved [31].

Now we show how to implement the convolution (58) in the CBS stage in our distributed setting. Once we can do this, there is no difficulty in doing decimation. We make the following assumptions to simplify our distributed algorithm.

- *Assumption 1*: The CBS filter has order $L-1 \le Q$. Since CBS is typically used for large arrays (large N), this assumption is reasonable.
- *Assumption 2*: There is an edge between each pair of adjacent nodes $p-1$ and p.

The method described in the following applies to CBS using any kinds of filters, including Capon-CBS [44]. With $\mathbf{y}$ and $\mathbf{H}$ defined as in (58), let $\mathbf{y}_0 \in \mathbb{C}^{Q-L+1}$ and $\mathbf{y}_p \in \mathbb{C}^Q$, $p = 1\ldots,P-1$ so that $\mathbf{y} = [\mathbf{y}_0^T\ \mathbf{y}_1^T\cdots\mathbf{y}_{P-1}^T]^T$. Since $L-1 \le Q$, we have

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{0,0} & \mathbf{0} & & \cdots & \mathbf{0} \\ \mathbf{H}_{1,0} & \mathbf{H}_{1,1} & \mathbf{0} & \cdots & \vdots \\ \mathbf{0} & \mathbf{H}_{2,1} & \mathbf{H}_{2,2} & \ddots & \\ \vdots & \ddots & \ddots & \ddots & \\ & & & & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{H}_{P-1,P-2} & \mathbf{H}_{P-1,P-1} \end{bmatrix}. \tag{63}$$

Here $\mathbf{H}_{0,0}$ is a $(Q-L+1)\times Q$ Toeplitz matrix with first row $[h(L-1)\ h(L-2)\cdots h(0)\ 0\cdots 0]$ and first column $[h(L-1)\ 0\cdots 0]^T$. Besides,

$$\mathbf{H}_{p,p-1} = \left[\begin{array}{c|cccc} & h(L-1) & h(L-2) & \cdots & h(1) \\ & 0 & h(L-1) & \cdots & h(2) \\ \mathbf{0} & \vdots & \ddots & \ddots & \vdots \\ & 0 & \cdots & 0 & h(L-1) \\ \hline \mathbf{0} & & \mathbf{0} & & \end{array}\right]$$

are $Q\times Q$ Toeplitz matrices, and $\mathbf{H}_{p,p}$ are $Q\times Q$ Toeplitz matrices with first row $[h(0)\ \ 0\cdots 0]$ and first column $[h(0)\ h(1)\cdots h(L-1)\ 0\cdots 0]^T$, for $p = 1,\ldots,P-1$. Thus, we have $\mathbf{y}_0 = \mathbf{H}_{0,0}\mathbf{x}_0$, which can be computed locally at node 0. Besides, since $L-1 \le Q$, we have

$$\mathbf{y}_p = \mathbf{H}_{p,p-1}\mathbf{x}_{p-1} + \mathbf{H}_{p,p}\mathbf{x}_p, \tag{64}$$

which can be computed locally at node p once node $p-1$ sends the last $L-1$ entries of $\mathbf{x}_{p-1}$, i.e., $x(pQ-L+1),\ldots,x(pQ-1)$, to node p, for $p = 1,\ldots,P-1$. This can be done without using AC because we assume that there is an edge between nodes $p-1$ and p. The communication cost per edge is $O(KL)$, where K is the number of snapshots.

If *no assumption* is made on the filter length L, then in the most general case, we can compute

$$y(i+L-1) = \mathbf{h}_i^T\mathbf{x} = \mathrm{AC}_p(\mathbf{h}_{i,p}^T\mathbf{x}_p), \tag{65}$$

where $\mathbf{h}_i^T$ is the ith row of $\mathbf{H}$, and $\mathbf{h}_{i,p} \in \mathbb{C}^Q$ is the subvector of $\mathbf{h}_i$ corresponding to node p, for $i = 0,\ldots,N-L$. The communication cost per edge is $O(K(N-L)I_{\rm ac})$.

B. Distributed Design of the Capon-CBS Filter

In [31], the CBS filter is a standard lowpass filter, such as the Parks-McClellan filter. In this case we simply assume all nodes have the filter coefficients available. It is beneficial to replace such a standard filter with a data-dependent filter based on the idea of Capon beamforming [44]. Among all the unknown DOAs, those falling in the stopband are treated as interferers (or jammers). We call this the Capon-CBS filter. This is nothing but a *sliding Capon beamspace generator*. This will provide optimal attenuation of the stopband signals.

Note that the output of this Capon-CBS filter will be used to estimate the DOAs in its passband. Thus, unlike in

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

traditional beamforming applications, this Capon-CBS filter should be designed to have a *flat passband*. Since the traditional Capon filter does not yield a flat passband around ω_0 although $H(e^{j\omega_0}) = 1$, we will use the robust Capon beamforming method reported in [46]. This is a method to ensure that $H(e^{j\omega}) \approx 1$ in a specified passband range, while at the same time optimally rejecting the interferers in the stopband. We call this CBS filter the *robust Capon beamspace filter* [44]. For purposes of computation [46], this filter will be expressed explicitly in terms of real and imaginary components: $[\mathrm{Re}\{\mathbf{h}^T\} \quad \mathrm{Im}\{\mathbf{h}^T\}]^T \triangleq \widetilde{\mathbf{h}}$ where $\mathbf{h} = [h^*(L-1) \;\; h^*(L-2) \;\; \cdots \;\; h^*(0)]^T$. The CBS filter coefficients $\widetilde{\mathbf{h}}$ are designed as a robust Capon beamspace generator, which is the solution to

$$\begin{aligned} \min_{\widetilde{\mathbf{h}}} \quad & \widetilde{\mathbf{h}}^T \widetilde{\mathbf{R}}_L \widetilde{\mathbf{h}} \\ \text{subject to} \quad & \widetilde{\mathbf{h}}^T \mathbf{a} \geq 1 \;\; \forall \mathbf{a} \in \mathcal{E}, \end{aligned} \tag{66}$$

where

$$\widetilde{\mathbf{R}}_L = \frac{1}{N-L+1} \sum_{i=0}^{N-L} \mathrm{E}[\widetilde{\mathbf{x}}_{L,i} \widetilde{\mathbf{x}}_{L,i}^T] \tag{67}$$

with $\widetilde{\mathbf{x}}_{L,i} = [\mathrm{Re}\{\mathbf{x}_{L,i}^T\} \; \mathrm{Im}\{\mathbf{x}_{L,i}^T\}]^T$ and

$$\mathbf{x}_{L,i} = [x(i) \; x(i+1) \; \cdots x(i+L-1)]^T. \tag{68}$$

Here, $\mathcal{E}$ is an $2L$-dimensional ellipsoid that covers the range of values of $\widetilde{\mathbf{a}}_L(\omega) = [\mathrm{Re}\{\mathbf{a}_L(\omega)^T\} \; \mathrm{Im}\{\mathbf{a}_L(\omega)^T\}]^T$ in the passband. The low-complexity algorithm given in [46] can be used to solve the problem.

Now we show how to compute $\widetilde{\mathbf{h}}$ in our distributed setting. We make the same *Assumptions 1* and *2* as in Sec. IV-A to simplify the algorithm. Then, with K snapshots, we estimate

$$\widehat{\widetilde{\mathbf{R}}}_L = \frac{1}{K(N-L+1)} \sum_{k=1}^{K} \sum_{i=0}^{N-L} \widetilde{\mathbf{x}}_{L,i}[k] \widetilde{\mathbf{x}}_{L,i}^T[k] \tag{69}$$

$$= \frac{P}{N-L+1} \, \mathrm{AC}_p(\widetilde{\mathbf{R}}_{L,p}), \tag{70}$$

where

$$\widetilde{\mathbf{R}}_{L,0} = \frac{1}{K} \sum_{k=1}^{K} \sum_{i=0}^{Q-L} \widetilde{\mathbf{x}}_{L,i}[k] \widetilde{\mathbf{x}}_{L,i}^T[k] \tag{71}$$

and

$$\widetilde{\mathbf{R}}_{L,p} = \frac{1}{K} \sum_{k=1}^{K} \sum_{i=pQ-L+1}^{(p+1)Q-L} \widetilde{\mathbf{x}}_{L,i}[k] \widetilde{\mathbf{x}}_{L,i}^T[k] \tag{72}$$

for $p = 1, \ldots, P-1$. Considering (68), we see that (71) can be computed locally at node 0 since it involves only $x(0), \ldots, x(Q-1)$. To compute (72), since it involves $x(pQ-L+1), \ldots, x((p+1)Q-1)$, node $p-1$ has to send the $L-1$ samples $x(pQ-L+1), \ldots, x(pQ-1)$ to node p. Then (72) can be computed at node p. Note that these samples to be sent are exactly the same as those we need to send for the distributed implementation of the CBS filter. In other words, there is no additional communication cost for the "convolution" part of CBS if we already did the "Capon design" of Capon-CBS. After each node obtains $\widehat{\widetilde{\mathbf{R}}}_L$ via AC in (70), the problem (66) can be solved locally at each node using the method in [46]. The total communication cost per edge for the design of the Capon-CBS filter is $O(KL + L^2 I_{\mathrm{ac}})$, where I_{ac} is the number of AC iterations.

C. Distributed DOA Estimation in Stage 2

For subspace-based methods such as MUSIC, root-MUSIC, and ESPRIT, we estimate the average of the $J \times J$ covariances of all the polyphase components [31]

$$\widehat{\mathbf{R}}_{\mathrm{ave}} = \frac{1}{KM} \sum_{k=1}^{K} \sum_{l=0}^{M-1} \mathbf{v}_l[k] \mathbf{v}_l^H[k], \tag{73}$$

where the polyphase components $\mathbf{v}_l$ are defined in (60). Then we only have to compute the signal and noise eigenvectors of (73), and the remaining steps just follow Secs. III-B and III-C. Note that (73) is similar to (3). The elements of each vector $\mathbf{v}_l$ are distributed among the P nodes in exactly the same way that the elements of $\mathbf{x}$ were distributed in (3). The eigenvectors of $\widehat{\mathbf{R}}_{\mathrm{ave}}$ can therefore be computed exactly as we computed eigenvectors of $\widehat{\mathbf{R}}_{\mathbf{xx}}$ in Sec. II-B. From these the DOAs can be estimated unambiguously as described in [31]. The summation over l in (73) leads to an increase in communication cost compared to element-space, but CBS offers a smaller computational complexity. It is a tradeoff between computation and communication.

FOCUSS introduced in Sec. III-D can also be used to estimate the DOAs in Stage 2. How we use FOCUSS with CBS is similar to the formulation of the Lasso problem with CBS, described in Sec. IV of [31]. We just replace $\mathbf{D}$ and $\mathbf{X}$ in (51) and (53) by $\mathbf{D}_{L,0} = [\mathbf{a}_J(M\bar{\omega}_1) \; \mathbf{a}_J(M\bar{\omega}_2) \cdots \mathbf{a}_J(M\bar{\omega}_{d_0})]$ and $\mathbf{V}_0 = \mathbf{D}_0 \mathbf{H} \mathbf{X}$, respectively. Here $\mathbf{D}_0$ is the decimation matrix as in (61), and $\{\bar{\omega}_1, \ldots, \bar{\omega}_{d_0}\}$ is the grid of frequencies within the passband of $H(e^{j\omega})$. If the grid is uniform, $d_0 \approx d/M$. Note that since the passband width is $2\pi/M$, the arguments $M\bar{\omega}_k$ span a range of 2π. The rows of $\mathbf{V}_0$ are distributed among the P nodes in the same way that the rows of $\mathbf{X}$ were distributed in (54). Hence, the FOCUSS iterations can be done using AC in a similar way. Significant reduction in computational complexity is obtained due to the smaller size of the dictionary $\mathbf{D}_{L,0}$. Moreover, the communication cost per edge is $O(d_0 K I_{\mathrm{ac}} I_{\mathrm{fo}})$, smaller than $O(dK I_{\mathrm{ac}} I_{\mathrm{fo}})$ for the element-space FOCUSS. Thus, among the distributed DOA estimation methods, FOCUSS gets more benefits when used with CBS.

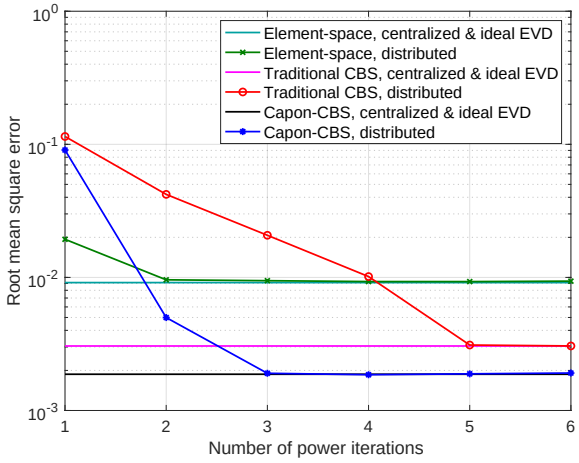
D. Simulations

We now present numerical examples to demonstrate the effectiveness of the proposed distributed CBS methods together with distributed DOA estimation algorithms in Sec. III. We again consider the same network (26) with $P = 6$, and use the finite-time average consensus (AC) method with 3 iterations as explained in Sec. III-A. The noise variance is $\sigma_e^2 = 1$.

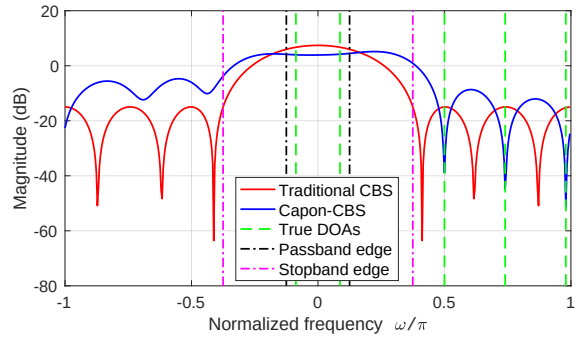
Example 1: Distributed root-MUSIC with distributed CBS. We show the DOA estimation performance of distributed root-MUSIC when it is used together with distributed CBS (both traditional and Capon). The results are also compared to element-space and the centralized counterparts. Again, the distributed power method is used to estimate eigenvectors of

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing



(a)



(b)

Fig. 4. Distributed root-MUSIC and centralized root-MUSIC for element-space, traditional CBS, and Capon-CBS. (a) RMSE of in-band DOA estimates. (b) Typical filter responses. The distributed and centralized algorithms result in the same filter for each system, so only one curve is plotted for each system.

the array output covariance for distributed algorithms, whereas ideal EVD is assumed for centralized algorithms. Each node is a ULA with $Q = 8$ sensors and they together form a 48-sensor ULA. For CBS methods, the filter length is $L = 9$, and the decimation ratio is $M = 4$. The traditional CBS filter is designed to be a lowpass Parks-McClellan filter [45], with passband edge $\pi/2M$ and stopband edge $3\pi/2M$. The Capon-CBS filter is the solution to Problem (66) with $\mathcal{E}$ designed as in [44] using $r = 10$ equally spaced samples of the array response in the passband. Note that the two assumptions $L - 1 \leq Q$ and that there is an edge between each pair of adjacent nodes $p - 1$ and p are satisfied, so the distributed CBS methods with lower communication costs can be used for both traditional and Capon CBS. There are 2 in-band sources (sources in the passband) with power -5 dB and DOAs $\theta = -5°, 5°$. There are 3 out-of-band sources (sources in the stopband) with power 15 dB and DOAs $\omega = 0.5\pi, 0.74\pi, 0.98\pi$. Each pair of the 5 sources has correlation coefficient 0.6, except that the two in-band sources are uncorrelated. We assume the number of DOAs is known. RMSEs of in-band DOA estimates for various number of power iterations are shown in Fig. 4(a). We use 100 snapshots and average 2000 Monte Carlo runs to get the plot. With only a few power iterations, all distributed algorithms converge to

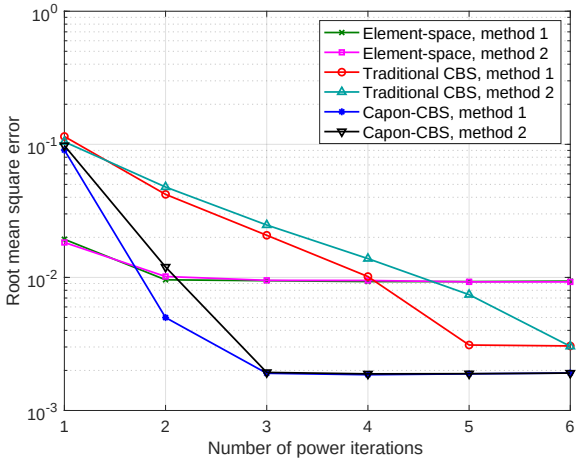


Fig. 5. RMSE of in-band DOA estimates for distributed root-MUSIC using the two distributed power methods. Method 1 is the traditional power method [8], where eigenvectors are updated sequentially. Method 2 is a modified version where all eigenvectors are updated in parallel.

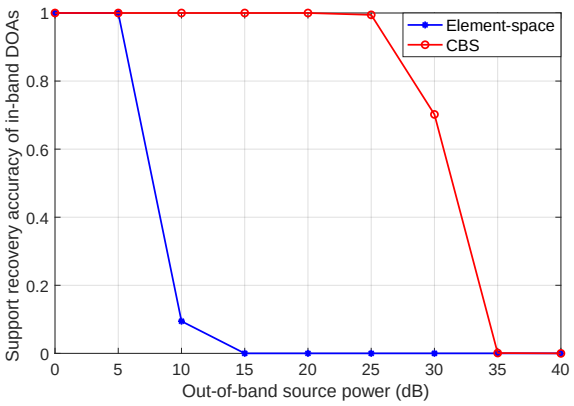
the centralized counterparts. Given enough number of power iterations, distributed traditional CBS and Capon-CBS perform significantly better than element-space. As shown in [44], in this harsh environment with powerful out-of-band sources correlated with in-band ones, centralized Capon-CBS has even smaller RMSE than centralized traditional CBS. This behavior is also true for distributed counterparts here. Fig. 4(b) also shows that the Capon-CBS filter indeed has a flat passband while suppressing the out-of-band sources. Capon-CBS filters keep a balance between attenuating interference and noise, which is the benefit of design based on data. The distributed and centralized algorithms result in the same filter for each of the two systems, traditional CBS and Capon-CBS, so only one curve is plotted for each system.

The results in Fig. 4 are obtained when we estimate eigenvectors of the array output covariance using the traditional distributed power method [8], where eigenvectors are updated sequentially (method 1). As mentioned in the end of Sec. II-B, we can also update the eigenvectors in parallel (method 2). For the same example, we compare the two distributed power methods in Fig. 5. We see that method 2 requires more iterations than method 1 to compute each eigenvector. This can be expected because we do not use the finally converged results of the first $q-1$ eigenvectors to update the qth eigenvector in method 2. Thus, we choose to use method 1 in all other examples in this paper.

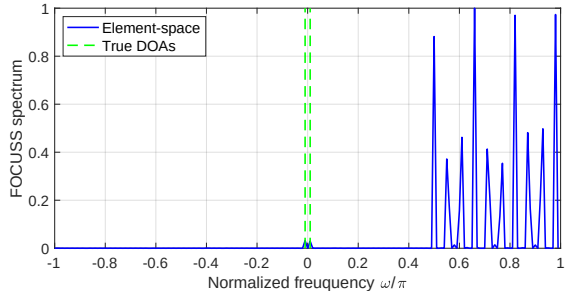
Example 2: Distributed FOCUSS with distributed CBS. Next we show the DOA estimation performance of distributed FOCUSS when it is used together with distributed CBS. The results are also compared to element-space and the centralized counterparts. For FOCUSS, the number of DOAs and their locations are estimated together. Specifically, after getting the estimate $\hat{\mathbf{Q}}$ in (50), we plot the *FOCUSS spectrum* $P(\bar{\omega}_i) = \sum_n |\hat{\mathbf{Q}}_{in}|^2$ for $1 \leq i \leq d$. Then, we declare that there is a source at $\bar{\omega}_i$ if there is a peak (local maximum) that is larger than a particular threshold: $P(\bar{\omega}_i) \geq \epsilon_{\text{fo}}$. Here we use $\epsilon_{\text{fo}} = 0.1$ (with the spectrum normalized to have a maximum value 1). Each node is a ULA with $Q = 16$

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

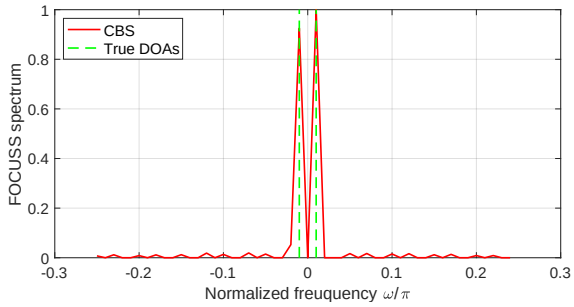
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing



(a)



(b)



(c)

Fig. 6. Distributed FOCUSS and centralized FOCUSS for element-space and CBS. (a) Support recovery accuracy (SRC) of in-band DOAs. Both distributed and centralized algorithms have the same SRC for each system, so only one curve is plotted for each system. (b)-(c) Typical FOCUSS spectra of element-space and CBS when the out-of-band source power is 15 dB.

sensors and they together form a 96-sensor ULA. For CBS, the decimation ratio is $M = 4$, and the filter is designed to be a lowpass Parks-McClellan filter [45] of length $L = 17$, with passband edge $\pi/2M$ and stopband edge $3\pi/2M$. A grid of $d = 200$ points uniform in ω is used for the potential DOAs. There are 2 in-band sources with power 1 and DOAs $\theta = -0.573°, 0.573°$. There are 10 out-of-band sources with identical power (which is varied in the experiment), with DOAs $\omega = 0.5\pi, 0.5\pi + \delta, 0.5\pi + 2\delta, \ldots, 0.98\pi$ with $\delta = 0.48\pi/9$. The in-band DOAs are exactly on the grid for simplicity. All sources are uncorrelated. Support recovery accuracy (SRC) of in-band DOAs for various out-of-band source powers are shown in Fig. 6(a). The SRC is defined as the probability of recovering the two and only two DOAs in the passband and without errors among all Monte Carlo runs. We use 100 snapshots and 2000 Monte Carlo runs to get the plot. CBS can tolerate more powerful out-of-band sources than element-space. More importantly, since distributed and centralized algorithms perform exactly the same for each of the two systems (i.e., element space and CBS), only one curve is plotted for each system. That is, the proposed distributed FOCUSS yields exactly the same solution as centralized FOCUSS. Typical FOCUSS spectra of element-space and CBS for a Monte Carlo run are shown in Fig. 6(b)-(c). For CBS, only the passband part is plotted. The two in-band sources can clearly be resolved with CBS FOCUSS, but not with element-space FOCUSS. In the latter the out-of-band sources are too powerful to allow the in-band sources to be resolved.

V. Distributed Spatial Smoothing

In this section, we propose distributed algorithms for spatial smoothing [33]. Spatial smoothing is a technique used for DOA estimation when there are coherent or correlated sources. Consider a ULA output $\mathbf{x}$ as in (1), which has covariance

$$\mathbf{R_{xx}} = \mathbf{A}\mathbf{R_{cc}}\mathbf{A}^H + \sigma_e^2\mathbf{I}, \tag{74}$$

where $\mathbf{R_{cc}} = \mathrm{E}[\mathbf{c}\mathbf{c}^H]$. When there are coherent sources, the rank of $\mathbf{R_{cc}}$ and hence the rank of $\mathbf{A}\mathbf{R_{cc}}\mathbf{A}^H$ will be less than the number of sources D. Therefore, the signal subspace, i.e., the column space of $\mathbf{A}$, cannot be fully identified from the EVD of $\mathbf{R_{xx}}$. To overcome this, one computes the spatially smoothed covariance

$$\mathbf{R}_{\mathrm{ss}} = \frac{1}{L_{\mathrm{ss}}} \sum_{i=0}^{L_{\mathrm{ss}}-1} \mathrm{E}[\mathbf{x}_{\mathrm{ss},i}\mathbf{x}_{\mathrm{ss},i}^H] \tag{75}$$

where L_{ss} is a parameter and

$$\mathbf{x}_{\mathrm{ss},i} = [x(i)\ x(i+1)\ \cdots x(i+N-L_{\mathrm{ss}})]^T. \tag{76}$$

It can be shown [33] that the reduced rank due to coherent sources can be fully restored back to D based on this spatial smoothing if $L_{\mathrm{ss}} \geq D$. Thus, more accurate DOA estimates can be obtained by applying subspace-based methods to $\mathbf{R}_{\mathrm{ss}}$, for at most $N - L_{ss}$ sources [33]. With this method we can therefore identify $D \leq N/2$ sources. Empirically, spatial smoothing also helps when sources are correlated but not necessarily coherent [48].

We now show how to do spatial smoothing in our distributed setting. The task here is to estimate the eigenvectors of $\widehat{\mathbf{R}}_{\mathrm{ss}}$ instead of $\widehat{\mathbf{R}}_{\mathbf{xx}}$ obtained from K snapshots and then estimate DOAs using, e.g., root-MUSIC. One may notice that both distributed design of the Capon-CBS filter and spatial smoothing involve the average of subarray covariance matrices. More precisely, $\widetilde{\mathbf{R}}_L$ in (67) and $\mathbf{R}_{\mathrm{ss}}$ in (75) are exactly the same if $L = N - L_{\mathrm{ss}} + 1$ and $\widetilde{\mathbf{x}}_{L,i}$ is replaced by $\mathbf{x}_{L,i}$. So actually, the method for dealing with $\widetilde{\mathbf{R}}_L$ can be used to deal with $\mathbf{R}_{\mathrm{ss}}$ if the assumptions (a) $(N - L_{\mathrm{ss}} + 1) - 1 \leq Q$ (b) that there is an edge between each pair of adjacent nodes $p-1$ and p, are satisfied. In this case, $\widehat{\mathbf{R}}_{\mathrm{ss}}$ can be computed via average consensus in a manner similar to (70). Then the eigenvectors of $\widehat{\mathbf{R}}_{\mathrm{ss}}$ can be computed locally at each node.

Case of unrestricted L_{ss}. If no assumption is made on L_{ss}, then we can modify the distributed power method in Sec. II-B

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

Algorithm 4 Distributed power iteration of spatial smoothing

1: **for** $i = 0$ to $L_{\rm ss} - 1$ **do**
2: **for** $k = 1$ to K **do**
3: $t_i[k] = P \cdot \mathrm{AC}_p(\mathbf{x}_{{\rm ss},i,p}^H[k]\mathbf{e}_{1,i,p}(n))$
4: **end for**
5: **for** $p = 0$ to $P - 1$ **do**
6: $\mathbf{q}_{i,p} = \frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_{{\rm ss},i,p}[k]t_i[k]$
7: **end for**
8: **end for**
9: **for** $m = 0$ to $L - 1$ **do**
10: **for** $p = 0$ to $P - 1$ **do**
11: $s_{m,p} = \frac{1}{L_{\rm ss}}\sum_{i:pQ\le i+m<(p+1)Q}[\mathbf{q}_i]_m$
12: **end for**
13: $[\mathbf{e}_1(n+1)]_m = P \cdot \mathrm{AC}_p(s_{m,p})$
14: **end for**
15: **return** $\mathbf{e}_1(n+1)$

to realize spatial smoothing. If we can show how to compute the power iterations

$$\mathbf{e}_1(n+1) = \widehat{\mathbf{R}}_{\rm ss}\mathbf{e}_1(n) \tag{77}$$

for the first eigenvector $\mathbf{e}_1$, then the remaining steps follow the same way as in Sec. II-B. The main idea is to clarify what computations can be done locally at each node p. The detailed procedure is presented in Algorithm 4 shown in the table. The vector $\mathbf{x}_{{\rm ss},i,p}$ is the subvector of $\mathbf{x}_{{\rm ss},i} \in \mathbb{C}^{N-L_{\rm ss}+1}$ stored at node p, and its length can vary from 0 to $N - L_{\rm ss} + 1$ for different i or p. Then, the vector $\mathbf{e}_{1,i,p}(n)$ is the subvector of $\mathbf{e}_1(n)$ with entries corresponding to $\mathbf{x}_{{\rm ss},i,p}$. The inner products $\mathbf{x}_{{\rm ss},i,p}^H[k]\mathbf{e}_{1,i,p}$ are computed locally at node p in parallel. The computations in Lines 6 and 11 are also done locally at each node in parallel. Finally, $\mathbf{q}_i = [\mathbf{q}_{i,0}^T \cdots \mathbf{q}_{i,P-1}^T]^T$, and the notations $[\mathbf{q}_i]_m$ and $[\mathbf{e}_1(n)]_m$ mean the mth entry of $\mathbf{q}_i$ and $\mathbf{e}_1(n)$, respectively. AC appears in Algorithm 4 in two places, Lines 3 and 13. Line 3 is the dominant operation for communication, so the total communication cost per edge for estimating D eigenvectors is $O(DKL_{\rm ss}I_{\rm ac}I_{\rm pm})$. This is $L_{\rm ss}$ times the cost of the basic distributed power method in Sec. II-B. This is a tradeoff for getting better DOA estimates when there are coherent or correlated sources and when $(N - L_{\rm ss} + 1) - 1 > Q$.

A numerical example is given to demonstrate the effectiveness of the proposed distributed spatial smoothing. We consider the same network (26) and use the finite-time AC method with 3 iterations as explained in Sec. III-A. We show the DOA estimation performance of distributed spatial smoothing used with root-MUSIC. Again, the distributed power method is used to estimate eigenvectors of the covariance for distributed algorithms, whereas ideal EVD is assumed for centralized algorithms. Each node is a ULA with $Q = 16$ sensors and they together form a 96-sensor ULA. There are 6 DOAs at angles $\theta = -10°, -6°, -2°, \ldots, 10°$. All sources have power 1, and each pair of sources have the same correlation coefficient ρ. The noise variance is $\sigma_e^2 = 1$. The parameter $L_{\rm ss} = 17$ is used for spatial smoothing. The assumption $(N-L_{\rm ss}+1)-1 \le Q$ is not satisfied, so Algorithm 4 is used. We assume the number of DOAs is known. The number of power iterations is $I_{\rm pm} = 5$. RMSEs of DOA estimates for various ρ are shown in Fig. 7. We use 500 snapshots and average 2000 Monte Carlo runs to get the plot. Spatial smoothing improves performance significantly, especially for the distributed case. Except for the coherent case ($\rho = 1$), RMSE of distributed spatial smoothing is almost the same as that of centralized spatial smoothing, though we use a small $I_{\rm pm} = 5$. When there is no spatial smoothing, or when $\rho = 1$, the distributed algorithm has larger RMSE than the centralized one because the covariance is closer to being rank-deficient, so errors due to finite $I_{\rm pm}$ are magnified.

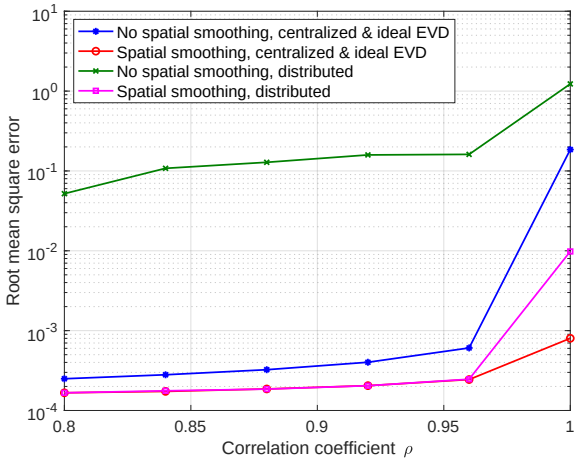


Fig. 7. RMSE of DOA estimates with and without spatial smoothing using distributed root-MUSIC and centralized root-MUSIC.

VI. Conclusion

In this paper we proposed distributed algorithms for several DOA estimation and beamforming methods, including spatial smoothing methods and the recently introduced convolutional beamspace methods. These algorithms are truly distributed in that a fusion center is not required. The novelties of the proposed algorithms lie in transforming problems at hand into steps where computing the average of some values across the network is the only step that involves data exchange among subarrays. Using average consensus with finite-time exact convergence as a subroutine, these distributed algorithms can achieve the same performance as centralized algorithms in just a few iterations. The effectiveness of the distributed algorithms is verified through simulations.

References

[1] J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proc. IEEE*, vol. 57, no. 8, pp. 1408–1418, 1969.
[2] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Trans. Antennas Propag.*, vol. 34, no. 3, pp. 276–280, Mar. 1986.
[3] B. D. Rao and K. V. S. Hari, "Performance analysis of Root-Music," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 12, pp. 1939–1949, 1989.
[4] R. Roy and T. Kailath, "ESPRIT – estimation of signal parameters via rotational invariance techniques," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 7, pp. 984–995, Jul. 1989.
[5] H. L. Van Trees, *Optimum array processing*. John Wiley & Sons, Inc., New York, 2002.
[6] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm," *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 600–616, 1997.

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSP.2021.3101015, IEEE Transactions on Signal Processing

[7] S. F. Cotter, B. D. Rao, Kjersti Engan, and K. Kreutz-Delgado, "Sparse solutions to linear inverse problems with multiple measurement vectors," *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2477–2488, 2005.

[8] A. Scaglione, R. Pagliari, and H. Krim, "The decentralized estimation of the sample covariance," in *Proc. Asilomar Conf. on Signal, Syst., Comput.*, 2008, pp. 1722–1726.

[9] S. X. Wu, H. Wai, L. Li, and A. Scaglione, "A review of distributed algorithms for principal component analysis," *Proc. of the IEEE*, vol. 106, no. 8, pp. 1321–1340, 2018.

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

[11] A. Sandryhaila, S. Kar, and J. M. F. Moura, "Finite-time distributed consensus through graph filters," in *Proc. IEEE Intl. Conf. Acoust., Speech, and Signal Process.*, 2014, pp. 1080–1084.

[12] S. Safavi and U. A. Khan, "Revisiting finite-time distributed algorithms via successive nulling of eigenvalues," *IEEE Signal Process. Lett.*, vol. 22, no. 1, pp. 54–57, 2015.

[13] W. Suleiman, M. Pesavento, and A. Zoubir, "Decentralized direction finding using partly calibrated arrays," in *European Signal Process. Conf. (EUSIPCO)*, 2013, pp. 1–5.

[14] A. Bertrand and M. Moonen, "Distributed adaptive estimation of covariance matrix eigenvectors in wireless sensor networks with application to distributed PCA," *Signal Process.*, vol. 104, pp. 120–135, 2014.

[15] A. Hassani, A. Bertrand, and M. Moonen, "Cooperative integrated noise reduction and node-specific direction-of-arrival estimation in a fully connected wireless acoustic sensor network," *Signal Process.*, vol. 107, pp. 68–81, 2015.

[16] M. Wax and T. Kailath, "Decentralized processing in sensor arrays," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 5, pp. 1123–1129, 1985.

[17] W. Suleiman and P. Parvazi, "Search-free decentralized direction-of-arrival estimation using common roots for non-coherent partly calibrated arrays," in *IEEE Intl. Conf. Acoust., Speech, and Signal Process.*, 2014, pp. 2292–2296.

[18] W. Suleiman, M. Pesavento, and A. M. Zoubir, "Performance analysis of the decentralized eigendecomposition and ESPRIT algorithm," *IEEE Trans. Signal Process.*, vol. 64, no. 9, pp. 2375–2386, 2016.

[19] B. Iancu and E. Isufi, "Towards finite-time consensus with graph convolutional neural networks," in *28th Eur. Signal Process. Conf. (EUSIPCO)*, 2021, pp. 2145–2149.

[20] D. Malioutov, M. Cetin, and A. S. Willsky, "A sparse signal reconstruction perspective for source localization with sensor arrays," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 3010–3022, Aug. 2005.

[21] B. N. Bhaskar, G. Tang, and B. Recht, "Atomic norm denoising with applications to line spectral estimation," *IEEE Trans. Signal Process.*, vol. 61, no. 23, pp. 5987–5999, 2013.

[22] G. Bienvenu and L. Kopp, "Decreasing high resolution method sensitivity by conventional beamformer preprocessing," in *Proc. IEEE Intl. Conf. Acoust., Speech, and Signal Process.*, Mar. 1984, pp. 714–717.

[23] K. M. Buckley and X. L. Xu, "Reduced-dimension beam-space broadband source localization: preprocessor design," in *Proc. SPIE, Adv. Algorithms and Architectures for Signal Process. III*, Feb. 1988, pp. 368–376.

[24] H. B. Lee and M. S. Wengrovitz, "Improved high-resolution direction-finding through use of homogeneous constraints," in *Fourth Annual ASSP Workshop on Spec. Est. and Modeling*, Aug. 1988, pp. 152–157.

[25] ——, "Resolution threshold of beamspace MUSIC for two closely spaced emitters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 9, pp. 1545–1559, Sep. 1990.

[26] X. L. Xu and K. M. Buckley, "Statistical performance comparison of MUSIC in element-space and beam-space," in *Proc. IEEE Intl. Conf. Acoust., Speech, and Signal Process.*, 1989, pp. 2124–2127.

[27] M. D. Zoltowski, G. M. Kautz, and S. D. Silverstein, "Beamspace root-MUSIC," *IEEE Trans. Signal Process.*, vol. 41, no. 1, pp. 344–364, Jan. 1993.

[28] G. Xu, S. D. Silverstein, R. H. Roy, and T. Kailath, "Beamspace ESPRIT," *IEEE Trans. Signal Process.*, vol. 42, no. 2, pp. 349–356, Feb. 1994.

[29] Z. Guo, X. Wang, and W. Heng, "Millimeter-wave channel estimation based on 2-D beamspace MUSIC method," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5384–5394, Aug. 2017.

[30] H. Zhao, N. Zhang, and Y. Shen, "Robust beamspace design for direct localization," in *IEEE Intl. Conf. Acoust., Speech and Signal Process.*, May 2019, pp. 4360–4364.

[31] P.-C. Chen and P. P. Vaidyanathan, "Convolutional beamspace for linear arrays," *IEEE Trans. Signal Process.*, vol. 68, pp. 5395–5410, 2020.

[32] P. P. Vaidyanathan and P.-C. Chen, "Convolutional beamspace for array signal processing," in *IEEE Intl. Conf. Acoust., Speech, and Signal Process.*, 2020.

[33] Tie-Jun Shan, M. Wax, and T. Kailath, "On spatial smoothing for direction-of-arrival estimation of coherent signals," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 33, no. 4, pp. 806–811, 1985.

[34] P. E. Gill, W. Murray, and M. H. Wright, *Practical optimization*. Academic Press, New York, 1981.

[35] O. Aberth, "Iteration methods for finding all zeros of a polynomial simultaneously," *Mathematics of computation*, vol. 27, no. 122, pp. 339–344, 1973.

[36] R. Merris, "Laplacian matrices of graphs: a survey," *Linear algebra and its applications*, vol. 197, pp. 143–176, 1994.

[37] G. W. Stewart, *Matrix Algorithms: Vol. II: Eigensystems*. SIAM, 2001.

[38] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–435, 1952.

[39] C.-Y. Chen and P. P. Vaidyanathan, "MIMO radar space-time adaptive processing using prolate spheroidal wave functions," *IEEE Trans. Signal Process.*, vol. 56, no. 2, pp. 623–635, 2008.

[40] P. Stoica and A. Nehorai, "MUSIC, maximum likelihood, and Cramer-Rao bound," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 5, pp. 720–741, May 1989.

[41] P. P. Vaidyanathan and P. Pal, "Direct-MUSIC on sparse arrays," in *2012 Intl. Conf. on Signal Process. and Commun. (SPCOM)*, 2012, pp. 1–5.

[42] ——, "Why does direct-MUSIC on sparse-arrays work?" in *2013 Asilomar Conf. on Signal, Syst., Comput.*, 2013, pp. 2007–2011.

[43] B. Ottersten, M. Viberg, and T. Kailath, "Performance analysis of the total least squares ESPRIT algorithm," *IEEE Trans. Signal Process.*, vol. 39, no. 5, pp. 1122–1135, 1991.

[44] P.-C. Chen and P. P. Vaidyanathan, "Sliding-Capon based convolutional beamspace for linear arrays," in *IEEE Intl. Conf. Acoust., Speech, and Signal Process.*, to appear.

[45] A. V. Oppenheim and R. W. Schafer, *Discrete-time signal processing*. Prentice Hall, 2010.

[46] R. G. Lorenz and S. P. Boyd, "Robust minimum variance beamforming," *IEEE Trans. Signal Process.*, vol. 53, no. 5, pp. 1684–1696, 2005.

[47] P. P. Vaidyanathan, *Multirate systems and filter banks*. Prentice Hall, Englewood Cliffs, N.J., 1993.

[48] J. S. Thompson, P. M. Grant, and B. Mulgrew, "Performance of spatial smoothing algorithms for correlated sources," *IEEE Trans. Signal Process.*, vol. 44, no. 4, pp. 1040–1046, 1996.

Po-Chih Chen (S'17) was born in 1993 and received the B.S. and M.S. degrees in electrical engineering and communication engineering from National Taiwan University (NTU), Taipei, Taiwan, in 2015 and 2017, respectively. He is currently pursuing the Ph.D. degree in electrical engineering at the California Institute of Technology (Caltech), Pasadena, CA. His research interests are in signal processing, array processing, sparse arrays, and distributed algorithms for arrays.

P. P. Vaidyanathan (S'80–M'83–SM'88–F'91) is the Kiyo and Eiko Tomiyasu Professor of Electrical Engineering at the California Institute of Technology. His research interests are in digital signal processing and machine learning. He is a Life Fellow of the IEEE, and recipient of the IEEE CAS Society Golden Jubilee Medal, and the Terman Award of the ASEE. He has received multiple awards for his papers, and for his teaching at Caltech, including the Northrop Grumman teaching prize. He is a recipient of the IEEE Gustav Robert Kirchhoff Award (2016), and the IEEE Signal Processing Society's Technical Achievement Award (2002), Education Award (2012), and Society Award (2016). He has been selected to received the EURASIP Athanasios Papoulis Award in 2021, and is a member of the U.S. National Academy of Engineering.

1053-587X (c) 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.
Authorized licensed use limited to: CALIFORNIA INSTITUTE OF TECHNOLOGY. Downloaded on August 04,2021 at 23:00:53 UTC from IEEE Xplore. Restrictions apply.