

The Decentralized Estimation of the Sample Covariance

Anna Scaglione, Roberto Pagliari and Hamid Krim
School of Electrical and Computer Engineering
Cornell University and North Carolina State University

Abstract—In this paper we consider the problem of estimating the eigenvectors of the sample covariance matrix of decentralized measurements in a distributed fashion. The need for a distributed scheme is motivated by the many moment based methods that resort to the covariance of the data to extract information from the measurements. For large sensor network, gathering the data at a central processor generates a communication bottleneck. Our algorithm is based on a combination of the so called power method, that is used to compute the eigenvectors, and the average consensus protocol, that is utilized to structure the information exchange into a gossiping protocol. Our work shows how a completely distributed scheme based on near neighbors communications is feasible, and applies the proposed method to the estimation of the direction of arrival of a signal source.

I. INTRODUCTION

One of the key problems in sensor networks is pruning the large amount of data measured, aggregating them into meaningful statistics of interest while reducing the data backlog.

In sensor networks' architectures with a single fusion center the information flows towards the same destination, facing a natural bottleneck. To overcome this problem several authors have argued in favor of computing along communication routes (see e.g. [5], [8], [2]), thereby trimming down the traffic as the data approach the central processor. But, if the network is large, classical routing schemes are difficult to manage, and optimal data aggregation has often a prohibitive overhead cost. Cluster based routing is simpler and mitigates the congestion issues, but the management of a hierarchical architecture remains non trivial, and network flow-based schemes are not well suited for changing topologies [14], [1]. These and other considerations make a compelling case for a fully distributed scheme to compute statistics, which does not assume any particular routing structure so as to be robust to mobility and sensor failure.

Second order statistics are particularly useful in a number of data analysis and compression methods. The data covariance matrix embeds a linear representation of the data involved in its so called principal components [6], that are the largest eigenvectors of the data sample covariance. The problem we are interested in is that of extracting in a decentralized fashion these largest eigenvectors, assuming that the data are measured by sensors which are locally connected via communication links.

A. Main Contribution

The protocol we propose is a fully decentralized method for estimating the eigenvectors of the sample covariance matrix of

sensor network data. The scheme does not need explicit routing of such data and the computation is performed in parallel by all sensors. In fact, every node in the network computes the corresponding entry of the k strongest eigenvectors of the sample covariance, starting from a scenario where the nodes have only a set of S local samples. Our scheme is based on combining the so called *power method* [16] with the network gossiping protocol called *average consensus* [15], [18]. There have been several applications of consensus gossiping for decentralized signal processing, see e.g. [10], [11], [17], [3], but to the best of our knowledge our is the first work that addresses the explicit computation of the eigenvectors of the sample covariance matrix. As an illustration of the use of this method, we consider the classical problem of decentralized beam-forming or, more specifically, estimating the direction of arrival (DoA) of a source [7]. At the end of our algorithm all nodes achieve an estimate of such angle. Compared to other distributed beam-forming schemes that have been proposed in literature ([9], [4], [13]) our method does not depend on the physical-mac layers, and the protocol is fully decentralized, i.e., it does not need any sort of hierarchical network management or cluster formation.

The paper is organized as follows: in Section II we describe the power method, and show how it is possible to derive the eigenvectors of the sample covariance matrix iteratively calculating a linear combination of inner products, i.e., linear functions of the decentralized set of data observed by the sensors. Section III briefly describes the average consensus protocol, which is a scheme to compute linear functions without a centralized control. In Section IV we indicate an implementation allowing to compute those eigenvectors distributedly. In Section V we present a simple example of application of this method, and simulation results are shown in Section VI. Section VII concludes the paper.

II. COMPUTATION OF THE COVARIANCE MATRIX EIGENVECTORS

We consider a network composed of N nodes, each one of them collects a set of S measurements of a common information source. These observations are assumed to be performed synchronously at absolute time instants (t_1, \dots, t_S) ; $x_i(t_s)$ denotes the observation made at the i -th sensor at time t_s , and $\mathbf{x}(t_s) = (x_1(t_s), \dots, x_N(t_s))^T \in \mathbb{C}^N$ is the vector containing all the observations at time t_s .

The covariance matrix is defined as

$$\mathbf{R} = E \{ (\mathbf{x} - E\{\mathbf{x}\})(\mathbf{x} - E\{\mathbf{x}\})^H \}. \quad (1)$$

An estimate of \mathbf{R} , given the S observation vectors defined above, is:

$$\hat{\mathbf{R}} = \frac{1}{S} \sum_{i=1}^S (\mathbf{x}(t_i) - \bar{\mathbf{x}})(\mathbf{x}(t_i) - \bar{\mathbf{x}})^H = \hat{\mathbf{U}} \hat{\mathbf{\Lambda}} \hat{\mathbf{U}}^H \quad (2)$$

In an ad-hoc network gathering all the S vectors and computing $\hat{\mathbf{R}}$ entails a considerable complexity. Motivated by this fact, we are interested in providing a tool to estimate in a completely decentralized fashion the eigenstructure of the sample covariance matrix of data acquired in a sensor network as well as a method that naturally allows to perform the projection of data onto such eigenvectors in a decentralized fashion. As explained next, our idea is based on the decomposition of the power method into a decentralized iterative protocol. In the following we assume that the sample covariance has no repeated non-zero eigenvalues.

A. The power method

The eigenvectors of $\hat{\mathbf{R}}$ can be derived by using a power-method as follows

$$\mathbf{v}(n+1) = \frac{\hat{\mathbf{R}}\mathbf{v}(n)}{\|\hat{\mathbf{R}}\mathbf{v}(n)\|} \quad (3)$$

where $\mathbf{v}(0)$ is an initial random vector in \mathbb{C}^N with a continuous distribution. This method converges to the maximum eigenvector of $\hat{\mathbf{R}}$ with probability 1, as long as the maximum eigenvalue of $\hat{\mathbf{R}}$ is strictly greater than the other eigenvalues and the vector $\mathbf{v}(0)$ has a non-zero component in the direction of the eigenvector associated to the largest eigenvalue.

Eliminating the normalization, for large n , we have that

$$\mathbf{v}(n+1) = \hat{\mathbf{R}}\mathbf{v}(n) = \hat{\mathbf{U}} \hat{\mathbf{\Lambda}}^n \hat{\mathbf{U}}^H \mathbf{v}(0) = \sum_{i=1}^N \hat{\lambda}_i^n \hat{\mathbf{u}}_i (\hat{\mathbf{u}}_i^H \cdot \mathbf{v}(0))$$

In the limit, the quantity $\frac{\mathbf{v}(n)}{\hat{\lambda}_1^n}$ tends to

$$\lim_{n \rightarrow \infty} \frac{\mathbf{v}(n)}{\hat{\lambda}_1^n} = (\hat{\mathbf{u}}_1^H \cdot \mathbf{v}(0)) \hat{\mathbf{u}}_1 + o\left(\left(\frac{\hat{\lambda}_2}{\hat{\lambda}_1}\right)^n \hat{\mathbf{u}}_2\right)$$

Therefore, if n is sufficiently large, the j -th component of the vector $\mathbf{v}(n)$, which is the local value at the j -th node, is the j -th component of the largest eigenvector of the covariance matrix $\hat{\mathbf{R}}$, scaled by the factor $\hat{\lambda}_1^n (\hat{\mathbf{u}}_1^H \cdot \mathbf{v}(0))$. This factor can be removed by normalizing the vector so that it has unit norm.

If the eigenvalues are all distinct and real, the eigenvectors are mutually orthogonal, and we have

$$(I - \hat{\mathbf{u}}_1 \hat{\mathbf{u}}_1^H) \hat{\mathbf{R}} = \sum_{i=2}^N \lambda_i \hat{\mathbf{u}}_i \hat{\mathbf{u}}_i^H$$

where eigenvectors $\hat{\mathbf{u}}_i, i = 1, \dots, N$ are associated to the eigenvalues $\lambda_1 > \lambda_2 > \dots > \lambda_N > 0$.

Algorithm 1 Derivation of the covariance matrix eigenvectors.

1. $\hat{\mathbf{R}}_1 = \frac{1}{S} \sum_{i=1}^S \mathbf{x}(t_i) \mathbf{x}(t_i)^H$
 2. $\hat{\mathbf{u}}_1 = \text{power_method}(\hat{\mathbf{R}}_1)$
 3. **For** $k = 2 : 1 : N$
 4. $\hat{\mathbf{R}}_k = (I - \hat{\mathbf{u}}_{k-1} \hat{\mathbf{u}}_{k-1}^H) \hat{\mathbf{R}}_{k-1}$
 5. $\hat{\mathbf{u}}_k = \text{power_method}(\hat{\mathbf{R}}_k)$
 6. **end for**
-

Based on this, a simple algorithm to compute all the N eigenvectors of a covariance matrix is given by A1.1 To decentralize the power method we propose a strategy to compute the iteration in Eq. (3) via the decentralized computation of inner products, which can be performed, as we will see later, through near-neighbors communications.

B. Derivation of the First Eigenvector

For simplicity let us assume that $E\{\mathbf{x}\} = \mathbf{0}$. Consider now the first two steps of algorithm A1.1. This part of the algorithm derives the eigenvector associated to the largest eigenvalue of the matrix $\hat{\mathbf{R}}$. The matrix $\hat{\mathbf{R}}_1 = \hat{\mathbf{R}}$ is completely determined by the observations made at the sensors, therefore, the recursive equation (3) can be expanded as follows:

$$\begin{aligned} \mathbf{v}_1(n+1) &= \hat{\mathbf{R}}\mathbf{v}_1(n) \\ &= \frac{1}{S} \sum_{i=1}^S \mathbf{x}(t_i) (\mathbf{x}(t_i)^H \cdot \mathbf{v}_1(n)) \\ &= \frac{N}{S} \sum_{i=1}^S \left(\frac{1}{N} \sum_{j=1}^N x_j(t_i) v_{1,j}(n) \right) \mathbf{x}(t_i). \end{aligned} \quad (4)$$

Hence, we can see that if node j has access to the S inner products $\mathbf{x}^H(t_s) \cdot \mathbf{v}_1(n)$, then it can calculate the j -th element $v_{1,j}(n+1)$ by simply computing the weighted sum of such inner products $\mathbf{x}^H(t_s) \cdot \mathbf{v}_1(n)$ with weights $x_j(t_s)$, for $s = 1, \dots, S$. The question is how can the node obtain the S inner products $\mathbf{x}^H(t_s) \cdot \mathbf{v}_1(n)$, for $s = 1, \dots, S$, distributedly via near-neighbors communications, considering that at each iteration the node has available only its local set of measurements $(x_j(t_1), \dots, x_j(t_S))$ and the j -th component of $\mathbf{v}_1(n)$. The method we propose to obtain $\mathbf{v}_1(n+1)$ is to employ S concurrent average consensus protocols, which entail the exchange of a data packet among neighbors with S computation/state variables at every iteration. Finally, for a sufficiently large $n = n^*$ the estimate of \mathbf{u}_1 is:

$$\hat{\mathbf{u}}_1 = \mathbf{v}_1(n^*) / \|\mathbf{v}_1(n^*)\|. \quad (5)$$

Hence, since each node computes only the corresponding entry of $\mathbf{v}_1(n^*)$ the last operation of the application of the power method consists in distributedly computing the norm $\|\mathbf{v}_1(n^*)\|$.

C. Computation of the k -th Eigenvector

Here we wish to extend the decomposition of the algorithm made above for the computation the remaining eigenvectors $\{\hat{\mathbf{u}}_2, \hat{\mathbf{u}}_3, \dots, \hat{\mathbf{u}}_N\}$ through the **For** cycle in algorithm A1.1.

In particular we want to show that, in a similar fashion, the decentralized computation of the entries of these eigenvectors requires the decentralized computation of inner products and norms of vectors whose elements are distributed among the nodes.

Consider the second iteration of that loop, i.e., the derivation of the second largest eigenvector of $\hat{\mathbf{R}}$. The eigenvector $\hat{\mathbf{u}}_2$ is the outcome of the power method based on the matrix $\hat{\mathbf{R}}_2 = (I - \hat{\mathbf{u}}_1 \hat{\mathbf{u}}_1^H) \hat{\mathbf{R}}_1$.

At the generic n -th iteration of the power-method algorithm, the update of vector $\mathbf{v}(n)$, given that vector $\mathbf{v}_2(0)$ has been randomly chosen, can be expressed as follows:

$$\begin{aligned} \mathbf{v}_2(n+1) &= (I - \hat{\mathbf{u}}_1 \hat{\mathbf{u}}_1^H) \hat{\mathbf{R}}_1 \mathbf{v}_2(n) \\ &= (I - \hat{\mathbf{u}}_1 \hat{\mathbf{u}}_1^H) \underbrace{\left(\frac{1}{S} \sum_{i=1}^S \mathbf{x}(t_i) (\mathbf{x}(t_i)^H \cdot \mathbf{v}_2(n)) \right)}_{\mathbf{v}'_2(n)} \\ &= \mathbf{v}'_2(n) - (\hat{\mathbf{u}}_1^H \cdot \mathbf{v}'_2(n)) \hat{\mathbf{u}}_1. \end{aligned} \quad (6)$$

Hence, each node j can calculate locally the corresponding component of $\mathbf{v}'_2(n)$ $v'_j(n)$ as long as it can access the inner products $(\mathbf{x}(t_i)^H \cdot \mathbf{v}_2(n))$; in addition, this time each node will have also to subtract also $\hat{u}_j (\hat{\mathbf{u}}_1^H \cdot \mathbf{v}'_2(n))$ to obtain $v_{2j}(n+1)$, adding extra complexity compared to the previous case since this requires the decentralized computation of $\hat{\mathbf{u}}_1^H \cdot \mathbf{v}'_2(n)$ as well.

The k -th eigenvector is then given by performing the following recursions that require an increasing number of projections to take place because:

$$\mathbf{v}_k(n+1) = \mathbf{v}'_k(n) - \sum_{i=1}^{k-1} (\hat{\mathbf{u}}_i^H \cdot \mathbf{v}'_k(n)) \hat{\mathbf{u}}_i. \quad (7)$$

As before, after a suitable number of iterations of the power method, each of the eigenvectors estimates is set to be $\hat{\mathbf{u}}_k = \mathbf{v}_k(n^*) / \|\mathbf{v}_k(n^*)\|$, where the norm has again to be computed in a decentralized fashion.

All of the above clarifies that the task of computing the eigenvectors of a sample covariance matrix can be solved if there is a mean of orchestrating the decentralized computation of the inner products and vector norms of interest; in fact, these are inner product and norms of vectors whose elements are known only by each corresponding node. Note also that in our decomposition of the problem, the sample covariance matrix is *never explicitly computed*, precisely to avoid the burden of sharing all observations among nodes. The missing element is the description of how this decentralized computation of inner products is performed. Hence, before fully describing our algorithm, we summarize the key aspects of the average consensus protocol which we use for the task of decentralized inner product calculation.

III. AVERAGE CONSENSUS PROTOCOL

Assume that the system of N sensor nodes is connected through a communication network that can be modeled as

a graph whose topology is represented by the corresponding Laplacian matrix \mathbf{L} whose elements are:

$$l_{ij} = \begin{cases} d_j, & i = j; \\ -1, & i \text{ communicates with } j. \\ 0 & \text{else} \end{cases} \quad (8)$$

where d_j is the degree of node j , i.e., the number of its neighbors. Let us omit the eigenvector index from the notation for simplicity. Considering the statement in Section II.B that each needs the quantity $\mathbf{x}^H(t_s) \cdot \mathbf{v}(n)$ to proceed to the next iteration, while the generic node j knows its own variables $(x_j^H(t_s), v_j(n))$ only. Let $z_j(0, t_s) = x_j^H(t_s) v_j(n)$. The inner product $\mathbf{x}^H(t_i) \cdot \mathbf{v}(n)$ is

$$\mathbf{x}^H(t_s) \cdot \mathbf{v}(n) = \sum_{j=1}^N x_j^H(t_s) v_j(n) = N \frac{1}{N} \sum_{j=1}^N z_j(0, t_s). \quad (9)$$

Hence, distributing the inner products $\mathbf{x}^H(t_i) \cdot \mathbf{v}(n)$ is equivalent to computing at all nodes the average $\bar{z}(0, t_s) \mathbf{1}$, where $\mathbf{1}$ is the vector of all 1's, and $\bar{z}(0, t_s) = \frac{1}{N} \sum_{j=1}^N z_j(0, t_s)$.

Let $\mathbf{W} = \mathbf{I} - \epsilon \mathbf{L}$. Distributed average consensus provides an estimate of this global average by performing at each the following update:

$$z_j(p+1, t_s) = W_{jj} z_j(p, t_s) + \sum_{k \in \mathcal{N}_j} W_{jk} z_k(p, t_s) \quad (10)$$

where $W_{jj} = 1 - \epsilon d_j$ and $W_{jk} = \epsilon$ if node k communicates with node j , 0 otherwise. Equivalently, the nodes perform the following network wide operation, by communicating with their neighbors only:

$$\mathbf{z}(p+1, t_s) = \mathbf{W} \mathbf{z}(p, t_s) = (\mathbf{I} - \epsilon \mathbf{L}) \mathbf{z}(p, t_s). \quad (11)$$

Eq. 10, or (11), $\hat{\mathbf{u}}$ converges to the sought average value as $p \rightarrow \infty$ iff $\mathbf{1}^T \mathbf{W} = \mathbf{1}^T$, $\mathbf{W} \mathbf{1} = \mathbf{1}$ and if the second largest eigenvalue of \mathbf{W} , $\lambda_2(\mathbf{W}) < 1$. In particular, for a finite number of iterations the average mean squared error is bounded by:

$$\frac{1}{N} \|\mathbf{z}(p, t_s) - \bar{z}(0, t_s) \mathbf{1}\|^2 \leq \lambda_2^p(\mathbf{W}) \|\mathbf{z}(0, t_s) - \bar{z}(0, t_s) \mathbf{1}\|^2.$$

Similarly, the computation of $\hat{\mathbf{u}}_i \mathbf{v}'_k(n)$ and also of $\|\mathbf{v}_k(n^*)\|^2$ can be performed by instances of the average consensus protocols that are initialized with the variables $z'_j(0) = \hat{u}_{ij} v'_{kj}(n)$ and $z''_j(0) = v_{kj}^2(n^*)$ respectively.

IV. IMPLEMENTATION

As a synopsis of what we just described, in this section we present the implementation of the sample covariance eigenvector estimation algorithm based on the decentralized power method, described in Sections II and III.

Algorithm 2 shows the derivation of the first eigenvector, while Algorithm 3 derives the remaining eigenvectors.

The function AC(arguments) implements the average consensus algorithm invoked by the previous two. The arguments could be one or many: in the latter case the same routine procedure will be applied in parallel to as many data as the arguments are, and the output will be a vector of the same size.

Algorithm 4 shows the implementation. The constant weight is fixed to the quantity $1/d_{\max}$ where d_{\max} is the maximum degree of the network, i.e., the maximum value on the diagonal of the Laplacian matrix \mathbf{L} .

Algorithm 2 First Eigenvector Estimation (algorithm performed at node j).

1. **Initialization**
 2. $\mathbf{x}_j = (x_j(t_1), \dots, x_j(t_S))$
 3. $\mathbf{v}_j = \text{rand ones}(S, 1)$
 4. **Recursion**
 5. **For** $i = 1 : 1 : n$
 6. $\mathbf{v}_j \leftarrow N \text{ AC}(\mathbf{v}_j \cdot \mathbf{x}_j^H)$;
 7. $\mathbf{v}_j \leftarrow 1/S \text{ sum}(\mathbf{v}_j \cdot \mathbf{x}_j) \text{ ones}(S, 1)$
 8. **end For**
 9. **Normalization**
 10. $u_{j1} \leftarrow \mathbf{v}_j(1)$
 11. $c \leftarrow u_{j1} \text{ conj}(u_{j1})$
 12. $c \leftarrow (N \text{ AC}(c))^{-5}$
 13. $u_{j1} \leftarrow u_{j1}/c$
-

Algorithm 3 k -th Eigenvector Estimation (algorithm performed at node j ; u_{jk} is the j -th component of eigenvector k).

1. **Initialization**
 2. $\mathbf{x}_j = (x_j(t_1), \dots, x_j(t_S))$
 3. $\mathbf{u}_j = (u_{j1}, \dots, u_{j(k-1)})$
 4. $\mathbf{v}_j = \text{rand ones}(S, 1)$
 5. **Recursion**
 6. **For** $i = 1 : 1 : n$
 7. $\mathbf{v}_j \leftarrow N \text{ AC}(\mathbf{v}_j \cdot \mathbf{x}_j^H)$;
 8. $\mathbf{v}_j \leftarrow 1/S \text{ sum}(\mathbf{v}_j \cdot \mathbf{x}_j) \text{ ones}(k-1, 1)$
 9. $\mathbf{v}_j(1) \leftarrow \mathbf{v}_j(1) - \text{sum}(N \text{ AC}(\mathbf{v}_j \cdot \mathbf{u}_j^H) \cdot \mathbf{u}_j)$
 10. $\mathbf{v}_j \leftarrow \mathbf{v}_j(1) \text{ ones}(S, 1)$
 11. **end For**
 12. **Normalization**
 13. $u_{jk} \leftarrow \mathbf{v}_j(1)$
 14. $c \leftarrow u_{jk} \text{ conj}(u_{jk})$
 15. $c \leftarrow (N \text{ AC}(c))^{-5}$
 16. $u_{jk} \leftarrow u_{jk}/c$
-

V. CASE STUDY: DISTRIBUTED DIRECTION OF ARRIVAL

As an application example, we consider the problem of estimating the direction of arrival of a signal observed by an array of sensors. The distributed direction of arrival (DDoA) estimation has several applications, including beamforming techniques and localization in wireless systems. We assume that the N sensors are placed as a uniform linear array (see Figure (1)) in known positions $r_i = (x_i, y_i) = ((i-1)d, 0)$, $i = 1, \dots, N$, and a source from an unknown angle relative to the direction of the array of sensors emits a zero mean stationary random signal with complex envelope $s(t) = Ae^{j\omega_0 t + \phi(t)}$ where $\phi(t)$ is random. The largest eigenvector is a sufficient

Algorithm 4 Average Consensus Protocol, AC(\cdot)

1. **Initialization Phase**
 2. Set_Clock(next_time);
 3. **Event Packet Received:**
 4. check_timestamp();
 5. add_entry();
 6. **Clock Fired:**
 7. **For** $i = 1 : \text{size}(\text{neighbor_table})$
 8. $\text{sum} += \text{weight} \cdot (\text{neighbor_table}(i).\text{value} - \text{value});$
 9. **end for**
 10. $\text{value} \leftarrow \text{value} + \text{sum};$
 11. Send_Avg_msg(ID, value, local_degree);
 12. **If** iterations \geq max
 13. exit;
 14. **else**
 15. Set_Clock(next_time);
 16. **end if**
-

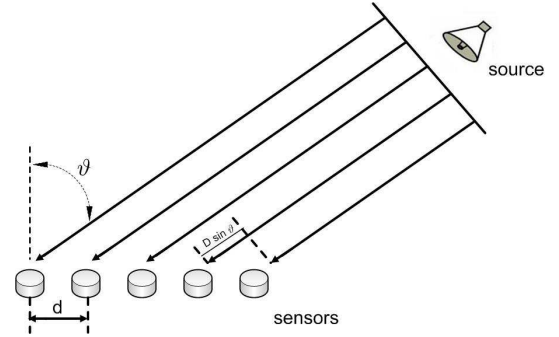


Fig. 1. Uniform Linear Array composed of $N = 5$ nodes, with inter-node distance d .

statistics for the angle of arrival estimation [12], and the problem at hand reduces to estimating \mathbf{u}_1 in fact, for the deployment in Figure (1), the received signal is

$$\mathbf{x}(t) = \mathbf{a}(\theta)s(t) + \mathbf{n}(t).$$

where $\mathbf{a}_{\text{ula}}(\theta) = g(\theta)[1 e^{-jkd\cos(\theta)} \dots e^{-j(L-1)kd\cos(\theta)}]$ is the so called steering vector, and $\mathbf{n}(t)$ is additive noise. In the following we assume $g(\theta) = 1$ for all the sensors. As well known, the covariance matrix is:

$$\mathbf{R} = A^2 \mathbf{a}(\theta) \mathbf{a}^H(\theta) + \sigma^2 \mathbf{I} \quad (12)$$

where $E\{\mathbf{n}(t)\mathbf{n}^H(t)\} = \sigma^2 \mathbf{I}$ is the noise covariance matrix. Clearly, $\lambda_{\max} = \lambda_1 = SA^2 + \sigma^2$ is the maximum eigenvalue and the associated eigenvector is $\mathbf{u}_{\max} = \mathbf{u}_1 = \mathbf{a}(\theta)/\|\mathbf{a}(\theta)\|$. Applying the method discussed the previous sections we can extract an estimate of \mathbf{u}_1 . The angle of arrival can be estimated computing in a decentralized fashion the $|\mathbf{a}(\theta)^H \cdot \mathbf{u}_{\max}|$ for several values of θ [12]. The estimate is:

$$\theta_{\text{source}} = \max_{\theta} \{|\mathbf{a}(\theta)^H \cdot \mathbf{u}_{\max}|\}. \quad (13)$$

VI. RESULTS

As an example, Fig.2 shows the outcome of Algorithm A1.2 applied to the estimation of the direction of arrival. In this

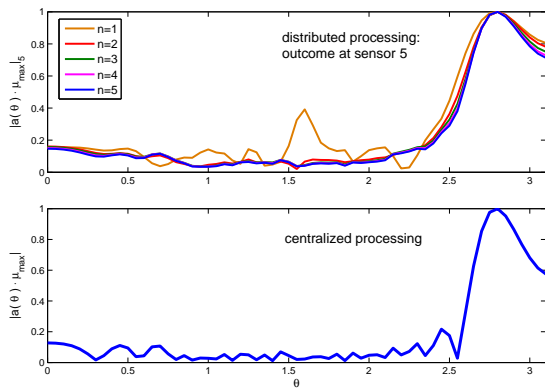


Fig. 2. DDoA estimation as a function of the number of recursions of the power method, with SNR = 0dB, $N = 20$ and $k = 20$ iterations of the average consensus.

case the network was composed of a uniform linear array of $N = 20$ sensors. The SNR was set to 0dB and $S = 100$ samples of the source were taken by the sensors.

In the picture is shown the estimate of the cost function $|\mathbf{a}(\theta)^H \cdot \mathbf{u}_{\max}|$ at node #5, for different number of power method iterations, and for $k = 20$ iterations of the average consensus, assuming the same initial data in all cases. The topology of the network allows communications between 4 neighbors. The function of a central processor that computes the eigenvector from the sample covariance is also shown as a reference. As we can see, even for a small number of iterations of the power method, the estimate of the angle of arrival is relatively accurate. This is due to the fact that the maximum eigenvalue, given by $\hat{\lambda}_1 = SA^2 + \hat{\sigma}^2$, is large because of the large number of samples S .

Fig.3 shows the average error on the estimate of the direction of arrival, as a function of the number of neighbors per node, ranging from 2 to 16. The error, computed as $E\{|\theta - \hat{\theta}|\}$, is averaged with respect to the sensors, over 1000 trials. In this case the topology is also a uniform linear array composed by $N = 20$ sensors, with SNR= 20dB, and $n = 5$ recursions of the power method. Not surprisingly we see a floor on the mean squared error, which is determined by the distortion due to the communication constraints, and the quantization of the parameter $\hat{\theta}$. To reduce the floor one needs to increase either the connectivity or the number of iterations of average consensus, and the number of bits required to represent the parameter $\hat{\theta}$.

VII. CONCLUSIONS

In this paper we have proposed a distributed protocol to estimate the eigenvectors of a covariance matrix. This protocol is based on a combination of two algorithms, the power method and the average consensus protocol. The result of this marriage is that the protocol retains the robustness, low overhead and simplicity of both algorithms. A decentralized computation of those eigenvectors is useful, because the covariance matrix embeds a linear representation of a set of data, and its first eigenvectors often represent a sufficient statistics for data

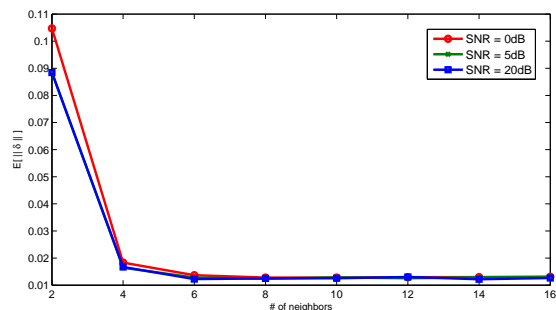


Fig. 3. DDoA average estimation error, with SNR = 20dB, $N = 20$, $n = 5$ recursions of the power method and $k = 20$ iterations of the average consensus protocol.

processing. We presented simulation results referred to specific application, the direction of arrival estimation, and we have shown by simulation that a distributed estimate of the angle is accurate and feasible.

REFERENCES

- [1] D. E. B. Krishnamachari and S. Wicker, "Modelling data-centric routing in wireless sensor networks," in *INFOCOM 2002*, New York, USA, 2002.
- [2] D. E. C. Intanagonwiwat, R. Govindan, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *IEEE/ACM Trans. on Networking*, 2003.
- [3] G. Cybenko, "Load balancing for distributed memory multiprocessors," *Journal of Parallel and Distributed Computing*, 1989.
- [4] H. P. H. Ochiai, P. Mitran and V. Tarokh, "Collaborative beamforming for distributed wireless ad hoc sensor networks," *IEEE Transaction on Signal Processing*, 2005.
- [5] N. M. J. Gao, L. Guibas and J. Hershberger, "Sparse data aggregation in sensor networks," in *IPSN 07*, Cambridge, Massachusetts, USA, 2007.
- [6] I. Jolliffe, "Principal component analysis," *Springer; 2nd ed. edition*, 2002.
- [7] H. Krim and M. Viberg, "Two decades of array signal processing algorithms," *IEEE Signal Processing Magazine*, 1996.
- [8] S. L. K.W. Fan and P. Sinha, "Structure-free data aggregation in sensor networks," *IEEE Trans. on Mobile Computing*, 2007.
- [9] A. P. L. Dong and H. Poor, "Cooperative beamforming for wireless ad hoc networks," in *Globecom 2007*, Washington DC, USA, 2007.
- [10] S. B. L. Xiao and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. International Conference on Information Processing in Sensor Networks*, Los Angeles, USA, 2005.
- [11] R. Olfati-Saber, "Distributed kalman filter with embedded consensus filters," in *Proc. 44th IEEE Conference on Decision and Control*, Sevilla, Spain, 2005.
- [12] M. V. P. Stoica, B. Ottersten and R. Moses, "Maximum likelihood array processing for stochastic coherent sources," *IEEE Transaction on Signal Processing*, vol. 44, no. 1, pp. 96–105, 1996.
- [13] G. B. R. Mudumbai and U. Madhow, "On the feasibility of distributed beamforming in wireless networks," *IEEE Transaction on Wireless Communications*, 2007.
- [14] R. Rajagopalan and P. Varshney, "Data-aggregation techniques in sensor networks: a survey," *Communications Surveys and Tutorials*, IEEE, 2006.
- [15] R. O. Saber and R. Murray, "Consensus protocols for networks of dynamic agents," in *American Control Conference*, 2003.
- [16] G. Stewart, "Matrix algorithms," *SIAM: Society for Industrial and Applied Mathematics*, 1998.
- [17] J. Tsitsikilis, "Problems in decentralized decision making and computation," *Ph.D. dissertation, MIT*, 1984.
- [18] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, 2004.