

# THE POWER-OJA METHOD FOR DECENTRALIZED SUBSPACE ESTIMATION/TRACKING

Sissi Xiaoxiao Wu<sup>†</sup>, Hoi-To Wai<sup>†</sup>, Anna Scaglione<sup>†</sup> and Neil A. Jacklin<sup>‡</sup>

<sup>†</sup>School of Electrical, Computer and Energy Engineering, Arizona State Univ., Tempe, AZ, USA

<sup>‡</sup>Northrop Grumman Mission Systems, McClellan, CA

## ABSTRACT

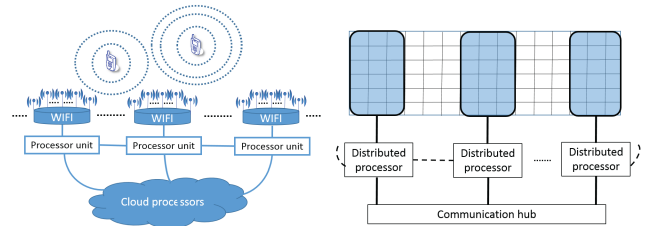
This work proposes a decentralized and adaptive subspace estimation method, called the Power-Oja (P-Oja) method. Existing decentralized subspace tracking algorithms have slow convergence rate or are unable to adapt to time varying statistics. To resolve these issues, the P-Oja method is developed by combining the power method with Oja's learning rule. Our key innovation lies on the design of a modified objective function with enhanced *spectral gap* property. This allows the P-Oja method to track the principal subspace more quickly with a finite number of samples. Interestingly, the resulting method coincides with the conventional Oja's learning rule in some special cases. To enable decentralized signal processing, we further demonstrate that the proposed method can be implemented by using a gossip algorithm. Our simulation results show that the proposed P-Oja outperforms the conventional Oja's method in terms of estimation accuracy, and the power method in terms of tracking performance. The effect of the communication graph on the tracking performance is also studied.

**Index Terms**— Massive arrays, spectrum sensing, subspace estimation, gossip algorithm.

## 1. INTRODUCTION

In this paper, we consider the problem of estimating cooperatively the signal subspace of transmitters that are measured by several separate antennas, forming a massive array. Our concept can be considered to be a possible architecture for distributed spectrum sensing in the fronthaul of a 5G network, where a virtualized physical layer, running over a Cloud Radio Access Network, will process measurements taken from several Remote Radio Heads, substituting the traditional processing made by 4G base station servers [1, Section 4.1]. A specific example of the application is depicted in Fig. 1 (Left) and a possible system architecture is provided in Fig. 1 (Right). The massive arrays (antennas) of sensors are used cooperatively to estimate the principal subspace of the signal of primary WiFi users in a non-synchronized manner. The secondary 5GPP users can project their signal onto the primary users' orthogonal subspace and thus reuse the spectrum without interfering with the WiFi users [2, 3]. Such an application requires a subspace estimation and tracking algorithm with high accuracy, low latency and adaptive to variation of the signals' statistics. Since the sampling data of the massive arrays is huge, and users may join or leave the channel randomly, the challenge in the design of this algorithm is dealing with the curse of data dimensionality.

Driven by the application above, this paper focuses on developing a decentralized subspace tracking algorithm. To this end, two classical algorithms for the task are the power method [4] and the Oja's learning rule [5]. The power method is a batch processing method with fast convergence, yet the method is non-adaptive and



**Fig. 1.** (Left) A WiFi network. (Right) Grouping antennas into sub-arrays with distributed processors for spectrum sensing.

has a high latency. On the other hand, the Oja's learning rule is an adaptive method derived from the stochastic gradient descent (SGD) method. The method is commonly used for tracking time varying statistics as the stochastic gradients are obtained from only one sample of the signal, yet it may suffer from slow convergence as it is essentially a first order optimization method.

Extensions to the classical algorithms above can be found in the literature. For example, our previous work [6] developed a decentralized version of the Oja's learning rule and analyzed its convergence; [7] considered a decentralized power method and [8] analyzed the convergence; [9] adopted similar principle to develop stochastic algorithms for related problems; [10, 11] performed convergence rate analysis for the algorithms. Recently, other authors [12] have considered using accelerated gradient techniques to speed up the convergence rate of the subspace estimation. To our knowledge, none of the previous works have considered the *tracking* power method.

This paper proposes the Power-Oja (P-Oja) method which integrates the power method with Oja's learning rule. Our algorithm is motivated from a modified objective function for principal subspace tracking with better convergence properties, whose optimal solution is the  $p$ -D principal subspace. We derive an SGD algorithm for tackling the stochastic optimization and draw connections between the power method and part of the SGD steps involved. These SGD steps are replaced by the power method subroutine to accelerate convergence. We note that the P-Oja method is a mini-batch method for which the batch size can be adjusted to trade-off between subspace tracking adaptivity and accuracy. We demonstrate that the P-Oja computations can be distributed to different processor units by employing the gossip protocol done in a similar fashion as [6, 7]. Our numerical results show that P-Oja demonstrates much better tracking performance than state-of-the-art methods.

**Notations** — We use the standard notations in this paper. For example,  $\|\cdot\|$  is the Euclidean norm,  $(\cdot)^H$  is the Hermitian transpose.  $\lambda_i(\mathbf{R})$  denotes the  $i$ th largest eigenvalue of  $\mathbf{R}$ .

## 2. PROBLEM STATEMENT

Consider a non-stationary stochastic process  $\mathbf{r}(t) \in \mathbb{C}^N$  with zero-mean and covariance matrix:

$$\mathbf{R}(t) = \mathbb{E}[\mathbf{r}(t)\mathbf{r}^H(t)] \in \mathbb{C}^{N \times N}. \quad (1)$$

This paper is concerned with tracking the top  $p$ -D subspace of  $\mathbf{r}(t)$  by tackling the non-convex, stochastic optimization:

$$\min_{\mathbf{U} \in \mathbb{C}^{N \times p}} f_t(\mathbf{U}) := \mathbb{E} \left[ \|\mathbf{r}(t) - \mathbf{U}\mathbf{U}^H \mathbf{r}(t)\|^2 \right], \quad \forall t \geq 1, \quad (2)$$

where we have relaxed the manifold constraint on  $\mathbf{U}$  which is a point on the Grassmannian.

Notice that the covariance matrix  $\mathbf{R}(t)$  can only be estimated from the samples  $\{\mathbf{r}(t)\}_{t \geq 1}$ . Let  $\mathcal{T} \subset \{1, 2, \dots\}$  be a sampling set. We define the corresponding sampled covariance as:

$$\hat{\mathbf{R}}(\mathcal{T}) := |\mathcal{T}|^{-1} \sum_{s \in \mathcal{T}} \mathbf{r}(s)\mathbf{r}^H(s). \quad (3)$$

In this paper, we will often work with the following stochastic approximation to the objective function  $f(\mathbf{U})$ :

$$\hat{f}(\mathbf{U}; \mathcal{T}_\tau) := \text{Tr} \left( \left( \mathbf{U}\mathbf{U}^H \mathbf{U}\mathbf{U}^H - 2\mathbf{U}\mathbf{U}^H \right) \hat{\mathbf{R}}(\mathcal{T}_\tau) \right), \quad (4)$$

where  $\mathcal{T}_\tau \subset \{1, 2, \dots\}$  is the set of observations made during the  $\tau$ th batch. If the stochastic process  $\mathbf{r}(t)$  is stationary for all  $t \in \mathcal{T}_\tau$ , then  $\mathbb{E}[\hat{f}(\mathbf{U}; \mathcal{T}_\tau)] = f_t(\mathbf{U})$ . When the batch set  $\mathcal{T}_\tau$  is large,  $\hat{f}(\mathbf{U}; \mathcal{T}_\tau)$  is a good approximation to the objective function  $f_t(\mathbf{U})$ .

Conventional algorithms for tackling (2) include the Oja's learning rule and the power method. However, the former is an adaptive algorithm which needs a fine step-size to track the right descent direction, while the latter is a batch algorithm, which requires estimating the covariance as a whole.

## 3. PROPOSED TRACKING ALGORITHM

The Power-Oja (P-Oja) method combines the power method with Oja's learning rule. This section focuses on the centralized implementation, while the decentralized implementation will be discussed in Section 4. To fix ideas, let us describe the power method and the Oja's learning rule as follows.

*Power method (PM)* — Let  $\mathcal{T}_\tau$  be the  $\tau$ th batch of samples. In the power method, we first generate a random vector as an initial point  $\hat{\mathbf{u}}^1(1, \tau)$  and perform this update at the  $\ell$ th power iteration:

$$\hat{\mathbf{u}}^1(\ell + 1, \tau) = \hat{\mathbf{R}}(\mathcal{T}_\tau) \hat{\mathbf{u}}^1(\ell, \tau), \quad \forall \ell \geq 1. \quad (5)$$

For some large  $L$ , the vector  $\hat{\mathbf{u}}^1(\tau) := \frac{\hat{\mathbf{u}}^1(L, \tau)}{\|\hat{\mathbf{u}}^1(L, \tau)\|}$  converges to the top eigenvector of  $\hat{\mathbf{R}}(\mathcal{T}_\tau)$ . To obtain the  $k$ th largest eigenvector of  $\hat{\mathbf{R}}(\mathcal{T}_\tau)$  for  $k \geq 2$ , we proceed in a similar manner as the first eigenvector. We generate a random vector as our initial point  $\hat{\mathbf{u}}^k(1, \tau)$ , from which we remove the  $(k-1)$  previously found eigenvectors by the following procedure:

$$\hat{\mathbf{u}}^k(\ell + 1, \tau) = \hat{\mathbf{R}}(\mathcal{T}_\tau) \hat{\mathbf{u}}^k(\ell, \tau) \quad \forall \ell = 1, \dots, L, \quad (6)$$

$$- \sum_{j=1}^{k-1} (\hat{\mathbf{u}}^j(\tau))^H \left( \hat{\mathbf{R}}(\mathcal{T}_\tau) \hat{\mathbf{u}}^k(\ell, \tau) \right) \hat{\mathbf{u}}^j(\tau),$$

$$\hat{\mathbf{u}}^k(\tau) := \hat{\mathbf{u}}^k(L, \tau) / \|\hat{\mathbf{u}}^k(L, \tau)\|. \quad (7)$$

The matrix  $\hat{\mathbf{U}}_{PM}(\tau) := [\hat{\mathbf{u}}^1(\tau) \hat{\mathbf{u}}^2(\tau) \dots \hat{\mathbf{u}}^p(\tau)]$  denotes the final solution found using the power method.

It is well known that the power method converges at a geometric rate [4]. The number of power iterations needed,  $L$ , is moderate. That said, when the process  $\mathbf{r}(t)$  is non-stationary, one may only take a small batch size, i.e.,  $|\mathcal{T}_\tau| \ll \infty$ . This results in a poor approximation for  $\hat{\mathbf{R}}(\mathcal{T}_\tau)$  to  $\mathbf{R}(t)$  and a degraded performance.

*Oja's learning rule* — Unlike the power method, the Oja's learning rule is an *adaptive* algorithm developed from the stochastic gradient descent (SGD) algorithm for (2) and is more suitable for the subspace tracking application. The Oja's learning rule works with one sample of  $\mathbf{r}(t)$  at a time. In particular, let  $\hat{\mathbf{U}}_{Oja}(t) \in \mathbb{C}^{N \times p}$  be an estimate of  $\mathbf{U}(t)$  at iteration  $t$ , we perform the updates:

$$\hat{\mathbf{U}}_{Oja}(t+1) = \hat{\mathbf{U}}_{Oja}(t) - \gamma_t \nabla \hat{f}(\hat{\mathbf{U}}_{Oja}(t); \{t\}), \quad (8)$$

where  $\gamma_t > 0$  is a step size and  $\nabla \hat{f}(\hat{\mathbf{U}}_{Oja}(t); \{t\})$  is the gradient of  $\hat{f}(\mathbf{U}; \{t\})$  taken at  $\mathbf{U} = \hat{\mathbf{U}}_{Oja}(t)$ , evaluated as:

$$\begin{aligned} \nabla \hat{f}(\hat{\mathbf{U}}(t), \{t\}) &= -2\mathbf{r}(t)\mathbf{r}^H(t)\hat{\mathbf{U}}(t) \\ &+ \mathbf{r}(t)\mathbf{r}^H(t)\hat{\mathbf{U}}(t)\hat{\mathbf{U}}^H(t)\hat{\mathbf{U}}(t) + \hat{\mathbf{U}}(t)\hat{\mathbf{U}}^H(t)\mathbf{r}(t)\mathbf{r}^H(t)\hat{\mathbf{U}}(t). \end{aligned} \quad (9)$$

Due to the non-convex nature of (2), the global convergence of Oja's learning rule has remained elusive. Most of the available results are focused on the special cases with stationary  $\mathbf{r}(t)$ . For example, the authors in [10] proved that when  $p = 1$  and  $\gamma_t = c/t$ , then (8) returns the top eigenvector of  $\mathbf{R}$  as  $t \rightarrow \infty$  at a sub-linear rate of  $\mathcal{O}(1/t)$ ; [5, Theorem 2] have proven if  $\sum_t \gamma_t = \infty$ ,  $\sum_t \gamma_t^2 < \infty$ , then (8) converges almost surely to the principal  $p$ -dimensional subspace for general  $p$ , yet the convergence rate is not given.

Nevertheless, the Oja's learning rule is often used when the process  $\mathbf{r}(t)$  is non-stationary. To avoid getting stuck at the previous subspaces, a heuristic is to set  $\gamma_t$  to be a small *constant* such that the newly observed samples are sufficiently represented.

### 3.1. The Power-Oja (P-Oja) method

We describe the Power-Oja (P-Oja) method in the following. A main feature of the proposed method is that it allows one to trade-off between the fast convergence of PM and the tracking ability in the Oja's learning rule.

To describe the P-Oja method, we let  $L \geq 1$  be a fixed parameter and consider the following modified stochastic approximation of the objective function in (2):

$$\hat{f}_{POja}(\mathbf{U}; \mathcal{T}) = \text{Tr} \left( \left( \mathbf{U}\mathbf{U}^H \mathbf{U}\mathbf{U}^H - 2\mathbf{U}\mathbf{U}^H \right) (\hat{\mathbf{R}}(\mathcal{T}))^L \right). \quad (10)$$

The function above differs from (4) by taking the  $L$ th power of the sampled covariance  $\hat{\mathbf{R}}(\mathcal{T})$ . There are several motivations to consider the modified objective (10).

First of all, it can be checked that  $\hat{\mathbf{R}}(\mathcal{T})$  has the same top  $p$ -dimensional principal subspace as that of  $(\hat{\mathbf{R}}(\mathcal{T}))^L$ . As such, minimizing  $\hat{f}_{POja}(\mathbf{U}; \mathcal{T})$  and  $\hat{f}(\mathbf{U}; \mathcal{T})$  yield the same optimal solution. Secondly, the  $L$ th powered sample covariance  $(\hat{\mathbf{R}}(\mathcal{T}))^L$  has a better *spectral gap* than  $\hat{\mathbf{R}}(\mathcal{T})$ , i.e.,

$$\frac{\sigma_p((\hat{\mathbf{R}}(\mathcal{T}))^L) - \sigma_{p+1}((\hat{\mathbf{R}}(\mathcal{T}))^L)}{\sigma_p((\hat{\mathbf{R}}(\mathcal{T}))^L)} > \frac{\sigma_p(\hat{\mathbf{R}}(\mathcal{T})) - \sigma_{p+1}(\hat{\mathbf{R}}(\mathcal{T}))}{\sigma_p(\hat{\mathbf{R}}(\mathcal{T}))},$$

as we notice that  $\sigma_p((\hat{\mathbf{R}}(\mathcal{T}))^L) = \sigma_p(\hat{\mathbf{R}}(\mathcal{T}))^L$ , where  $\sigma_p(\mathbf{R})$  denotes the  $p$ th largest singular value of  $\mathbf{R}$ . As seen in the analysis of [10], the size of the spectral gap  $\sigma_p((\hat{\mathbf{R}}(\mathcal{T}))^L) - \sigma_{p+1}((\hat{\mathbf{R}}(\mathcal{T}))^L)$

is an important factor in determining the convergence speed of subspace estimation/tracking algorithms.

To reduce the variance of the top  $p$ -dimensional subspace of  $\hat{\mathbf{R}}(\mathcal{T})$  and thus the bias from  $\mathbf{R}(\mathcal{T})$  for the  $L$ th powered sample covariance  $(\hat{\mathbf{R}}(\mathcal{T}))^L$ , we take a batch processing approach in the P-Oja method. Specifically, we set a batch size to  $T$  and define  $\mathcal{T}_1 = \{1, \dots, T\}$ ,  $\mathcal{T}_2 = \{T+1, \dots, 2T\}$ , .... We then track the subspace in a batch by batch manner. Notice that in doing so we assume that the stochastic process  $\mathbf{r}(t)$  is stationary for the samples within a batch. Now, let us apply the SGD method on the modified objective (10). Considering the  $\tau$ th batch, we observe that the gradient of  $\hat{f}_{\text{POja}}(\cdot; \cdot)$  at  $\hat{\mathbf{U}}_{\text{POja}}(\tau)$  can be evaluated as

$$\begin{aligned} \nabla \hat{f}_{\text{POja}}(\hat{\mathbf{U}}_{\text{POja}}(\tau); \mathcal{T}_\tau) = & \quad (11) \\ & -2(\hat{\mathbf{R}}(\mathcal{T}_\tau))^L \hat{\mathbf{U}}_{\text{POja}}(\tau) + (\hat{\mathbf{R}}(\mathcal{T}_\tau))^L \hat{\mathbf{U}}_{\text{POja}}(\tau) \hat{\mathbf{U}}_{\text{POja}}^H(\tau) \hat{\mathbf{U}}_{\text{POja}}(\tau) \\ & + \hat{\mathbf{U}}_{\text{POja}}(\tau) \hat{\mathbf{U}}_{\text{POja}}^H(\tau) (\hat{\mathbf{R}}(\mathcal{T}_\tau))^L \hat{\mathbf{U}}_{\text{POja}}(\tau). \end{aligned}$$

Importantly, we notice that  $(\hat{\mathbf{R}}(\mathcal{T}_\tau))^L \hat{\mathbf{U}}_{\text{POja}}(\tau)$  is akin to performing  $L$  rounds of the power iterations on  $\hat{\mathbf{U}}_{\text{POja}}(\tau)$ . As such, we propose to replace this by the power method described in Eq. (5)–(7). In particular, let us denote the output from Eq. (5)–(7) as:

$$\bar{\mathbf{U}}_{PM}(\tau) = \text{PM}(\{\mathbf{r}(s)\}_{s \in \mathcal{T}_\tau}; \hat{\mathbf{U}}_{\text{POja}}(\tau); L), \quad (12)$$

where we used  $\{\mathbf{r}(s)\}_{s \in \mathcal{T}_\tau}$  to form  $\hat{\mathbf{R}}(\mathcal{T}_\tau)$ , the subspace found in the previous iteration  $\hat{\mathbf{U}}_{\text{POja}}(\tau)$  is used to initialize the power iterations for each dimension of the subspace and the power iteration (7) terminates after  $L$  recursions. We expect that  $\bar{\mathbf{U}}_{PM}(\tau) \approx (\hat{\mathbf{R}}(\mathcal{T}_\tau))^L \hat{\mathbf{U}}_{\text{POja}}(\tau)$ .

Finally, the P-Oja method is given by the following iterations:

$$\hat{\mathbf{U}}_{\text{POja}}(\tau+1) = \hat{\mathbf{U}}_{\text{POja}}(\tau) - \gamma_\tau \hat{\nabla} \hat{f}_{\text{POja}}(\hat{\mathbf{U}}_{\text{POja}}(\tau); \mathcal{T}_\tau), \quad (13)$$

where  $\hat{\nabla} \hat{f}_{\text{POja}}$  is the approximated gradient, evaluated as:

$$\begin{aligned} \hat{\nabla} \hat{f}_{\text{POja}}(\hat{\mathbf{U}}_{\text{POja}}(\tau); \mathcal{T}_\tau) = & \bar{\mathbf{U}}_{PM}(\tau) \hat{\mathbf{U}}_{\text{POja}}^H(\tau) \hat{\mathbf{U}}_{\text{POja}}(\tau) \quad (14) \\ & + \hat{\mathbf{U}}_{\text{POja}}(\tau) \hat{\mathbf{U}}_{\text{POja}}^H(\tau) \bar{\mathbf{U}}_{PM}(\tau) - 2\bar{\mathbf{U}}_{PM}(\tau). \end{aligned}$$

The P-Oja method is parameterized by  $L$  and  $T$ , where  $T$  controls the variance in the sampled covariance  $\hat{\mathbf{R}}(\mathcal{T}_\tau)$  and  $L$  corresponds to the acceleration given by the power method subroutine. It is worth noting that when  $p = 1$ ,  $T = 1$ ,  $L = 1$ , the P-Oja method is reduced into the Oja's learning rule.

*Remark:* In order to adopt the unconstrained SGD algorithm, the conventional Oja's method does not employ any unitary constraint on the subspace  $\mathbf{U}$ . Hence, even if we can orthogonalize the resulting subspace, there is no guarantee for a rank- $p$  subspace. In this work, since we propose to search the descent direction by the PM, if the samples in the batch is sufficient, we are very likely to obtain a rank- $p$  subspace. This is a by-product of the proposed P-Oja method.

#### 4. DECENTRALIZED SUBSPACE TRACKING

In this section, we show that the proposed P-Oja method can be implemented in a decentralized manner. To this end, we denote the communication network between  $M$  processor units as an undirected graph  $G = (V, E)$  such that  $V = \{1, \dots, M\}$  and  $E \subseteq V \times V$ . The graph is assumed to be sparse (e.g., via the wiring of the processor units and the communication hub in Fig. 1) and connected. Furthermore, there is a doubly stochastic matrix  $\mathbf{W}$  associated with

$G$  such that  $[\mathbf{W}]_{ij} = 0$  if and only if  $(i, j) \notin E$ . Each processor unit locally processes its subarray's sampling data, and meanwhile exchanges information with its neighbors in the graph. Armed with this architecture, we shall demonstrate how the power method (5)–(7) can be implemented in a decentralized manner. For brevity, we only consider the case of  $p = 1$ .

Similar to [7], we partition the vectors  $\mathbf{r}(t) = [\mathbf{r}_1(t); \dots; \mathbf{r}_M(t)]$  and  $\hat{\mathbf{u}}(\ell, \tau) = [\hat{\mathbf{u}}_1(\ell, \tau); \dots; \hat{\mathbf{u}}_M(\ell, \tau)]$ , where  $\mathbf{r}_i(t) \in \mathbb{C}^{\frac{N}{M}}$  is the receive signal at subarray  $i$  and  $\hat{\mathbf{u}}_i(\ell, \tau) \in \mathbb{C}^{\frac{N}{M}}$  is the estimated data stored and computed in processor unit  $i$ . Let  $z_i^0 := \mathbf{r}_i^H(t) \hat{\mathbf{u}}_i(\ell, \tau)$  and notice that  $z_i^0$  can be computed locally at processor unit  $i$ . The power iteration (5) can be expressed as:

$$\sum_{s \in \mathcal{T}_\tau} \mathbf{r}(s) \mathbf{r}^H(s) \hat{\mathbf{u}}(\ell, \tau) = \sum_{s \in \mathcal{T}_\tau} \mathbf{r}(s) \left( \sum_{i=1}^M z_i^0 \right) \quad (15)$$

In particular, the summation  $\sum_{i=1}^M z_i^0$  can be obtained by the gossip recursion:

$$z_i^{k+1} = \sum_{j=1}^M W_{ij} z_j^k, \quad k = 0, 1, \dots, K. \quad (16)$$

We denote  $z_i^K$  as the  $i$ th output of the subroutine  $\text{AC}^i(\{z_j^0\}_{j=1}^M; K)$ . Using linear algebra, it can be verified that

$$\|\mathbf{z}^K - \bar{\mathbf{z}}\mathbf{1}\| \leq |\lambda_2(\mathbf{W})|^K \|\mathbf{z}_0 - \bar{\mathbf{z}}\mathbf{1}\|, \quad (17)$$

where  $\mathbf{z}^K = [z_1^K; \dots; z_M^K]$  and  $\bar{\mathbf{z}} = \sum_{i=1}^M z_i^0 / M$ . In particular, when  $K$  is sufficiently large, then  $\sum_{j=1}^M z_j^0 \approx M \cdot \text{AC}^i(\{z_j^0\}_{j=1}^M; K)$  for all  $i$  and the convergence rate is geometric in  $K$  [13]. Consequently, we can replace the power iteration (5) by:

$$\hat{\mathbf{u}}_i(\ell+1, \tau) = \frac{1}{|\mathcal{T}_\tau|} \sum_{s \in \mathcal{T}_\tau} \mathbf{r}_i(s) (M \cdot \text{AC}^i(\{z_j^0\}_{j=1}^M; K)), \quad (18)$$

which can be computed locally at the  $i$ th distributed processor. For general  $p > 1$ , we observe that the inner product  $(\hat{\mathbf{u}}^j(\tau))^H \hat{\mathbf{u}}^k(1, \tau)$  in (6), the  $\ell_2$ -norm  $\|\hat{\mathbf{u}}^k(L, \tau)\|^2$  can be handled using a similar technique; see [6, 7] for details. More formally, given a transition matrix  $\mathbf{W}$  and the number of gossip iteration  $K$ , we can denote the output of the decentralized power processor by

$$\bar{\mathbf{U}}_{PM}^i(\tau) = \text{PM}^i(\{\mathbf{r}(s)\}_{s \in \mathcal{T}_\tau}, \hat{\mathbf{U}}_{\text{POja}}(\tau), \mathbf{W}, K, L) \in \mathbb{C}^{\frac{N}{M} \times p}, \quad (19)$$

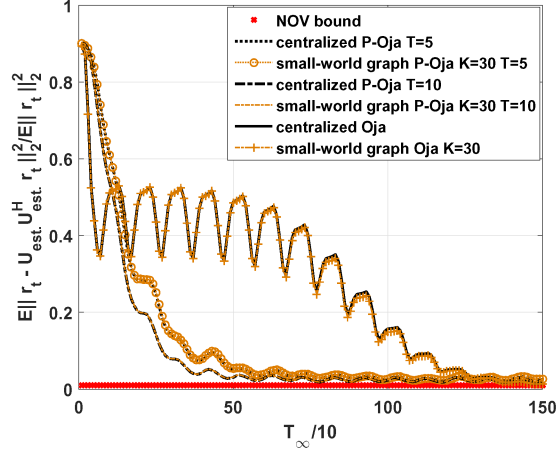
where  $\hat{\mathbf{U}}_{\text{POja}}(\tau) := [\hat{\mathbf{U}}_{\text{POja}}^1(\tau); \dots; \hat{\mathbf{U}}_{\text{POja}}^M(\tau)]$ .

Finally, we describe the distributed computation of the approximate gradient update (14). This can be achieved by dividing the gradient  $\hat{\nabla} \hat{f}_{\text{POja}}(\hat{\mathbf{U}}_{\text{POja}}(\tau); \mathcal{T}_\tau)$  into  $M$  matrices, each expressed by:

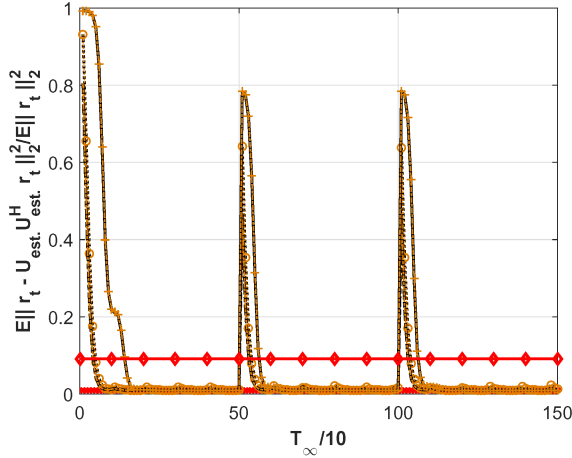
$$\begin{aligned} \hat{\nabla} \hat{f}_{\text{POja}}^i(\hat{\mathbf{U}}_{\text{POja}}(\tau); \mathcal{T}_\tau) = & \bar{\mathbf{U}}_{PM}^i(\tau) (\hat{\mathbf{U}}_{\text{POja}}^H(\tau) \hat{\mathbf{U}}_{\text{POja}}(\tau)) \quad (20) \\ & + \hat{\mathbf{U}}_{\text{POja}}^i(\tau) (\hat{\mathbf{U}}_{\text{POja}}^H(\tau) \bar{\mathbf{U}}_{PM}^{1:M}(\tau)) - 2\bar{\mathbf{U}}_{PM}^i(\tau), \end{aligned}$$

we also denote  $\bar{\mathbf{U}}_{PM}^{1:M}(\tau) = [\bar{\mathbf{U}}_{PM}^1(\tau); \dots; \bar{\mathbf{U}}_{PM}^M(\tau)]$ . Like [7], in the above, the matrices  $\hat{\mathbf{U}}_{\text{POja}}^H(\tau) \hat{\mathbf{U}}_{\text{POja}}(\tau)$  and  $\hat{\mathbf{U}}_{\text{POja}}^H(\tau) \bar{\mathbf{U}}_{PM}^{1:M}(\tau)$  can be computed in a decentralized manner. For example, we observe that  $\hat{\mathbf{U}}_{\text{POja}}^H(\tau) \bar{\mathbf{U}}_{PM}^{1:M}(\tau)$  equals to the summation  $\sum_{i=1}^M \mathbf{V}_i$  where  $\mathbf{V}_i = (\hat{\mathbf{U}}_{\text{POja}}^i(\tau))^H \bar{\mathbf{U}}_{PM}^i(\tau)$ .

We remark that the message exchanged during the gossip exchange stage within the power method (18) is only a scalar; while the message exchanged in the approximate gradient (20) is a  $p \times p$  matrix.



**Fig. 2.** The normalized objective value (NOV) for a constant 2-D signal space.



**Fig. 3.** The NOV for a variant 1-D signal space. The legend is the same as that in Fig. 1, except for the number of gossip iterations is now  $K = 10$ . The diamond-marked curve is the NOV for the conventional power method.

#### 4.1. The network topology

The performance of the proposed decentralized P-Oja method depends on the accuracy of the gossip protocol. In practice, it is crucial for us to choose a proper network topology. This can be seen in (17) as the convergence rate of the gossip protocol depends on the second largest eigenvalue of  $\mathbf{W}$ . On the other hand, considering the physical location of the processor units, it is more economical to connect the nearby units with a higher probability while the far-apart units with a lower probability. We propose to use the small-world graph with a proper rewiring probability. In particular, we adopt the optimal constant weights [14] to construct the transition matrix, i.e.,

$$\mathbf{W} = \mathbf{I} - \frac{2}{\lambda_1(\mathbf{L}) + \lambda_{N-1}(\mathbf{L})} \mathbf{L},$$

where  $\mathbf{L}$  is the Laplacian matrix.

## 5. NUMERICAL SIMULATION & CONCLUSIONS

In this section, we verify the P-Oja method by numerical simulations. We consider a massive array with  $N = 256$  antennas, which continuously receives 1500 samples of signals and tries to use different methods to estimate the subspace. The signal-to-noise ratio (SNR) is set to be 20dB. The massive array is grouped to  $M = 64$  subarrays, each equipped with four antennas. The processor units for the subarrays form a graph with 64 nodes. In this simulation, we adopt a degree-6 small-world graph with rewiring probability 0.2. We calculate the normalized objective value (NOV) of (2) for both the estimation and tracking processes under different scenarios. In all simulations, the power iteration is  $L = 20$ . The step size  $\gamma_t$  for the Oja's learning rule is set to be  $5 \times 10^{-4}$  and for P-Oja is set to be  $0.01T$  in Fig. 2 and  $0.04T$  in Fig. 3, where  $T$  is the number of samples in one batch.

In Fig. 2, we compare the NOV for different methods when  $\mathbf{r}(t)$  is stationary over the 1500 samples tested. In particular, we consider a massive *linear* array, sensing two line-of-sight users with different angles of arrivals, forming a 2-D signal space. We test different batch size  $T = 5$  and 10. For comparison convenience, we set the x-axis as the sample index divided by 10, and the y-axis is the NOV. In Fig. 2, we show both centralized and decentralized algorithms. The NOV bound is obtained by using the true subspace. We see that, as  $T$  increases, the convergence rate increases; e.g., see the curves for  $T = 5$  and  $T = 10$ . When  $K$  is sufficiently large ( $K = 30$  in this case), the decentralized performance will approach the centralized one. The numerical results show that the P-Oja method converges much faster than the Oja's method, and the decentralized algorithms work well under the chosen graph.

In Fig. 3, we compare the tracking performance for different methods with a non-stationary  $\mathbf{r}(t)$ . We still consider the linear array. However, due to the movement of the users, compared to the first 500 samples, the direction of arrivals changes 0.3 degree and 0.6 degree for the next 500 samples and the last 500 samples, respectively. Hence, we have in total three groups of 500 samples, each of which has the same covariance matrix. Thus, the NOV bounds are evaluated by 500 samples with the same statistic. We see that, as  $T$  increases, the convergence rate increases; e.g., see the curves for  $T = 5$  and  $T = 10$ . When  $K = 10$ , the decentralized and centralized methods coincide with each other. We also plot the conventional, centralized power method as a benchmark, for which the covariance is evaluated by 1500 samples. Apparently, the power method cannot track the change of the covariance. Our simulation results demonstrate that the proposed P-Oja can both track the change of statistic, but converges much faster than the conventional Oja method.

The conventional subspace estimation methods either estimate by treating the received samples as a whole, e.g., the power method, or update subspace sample by sample. The former provides a good estimation performance for stationary signals, however results in a large latency and cannot handle non-stationary signals; the latter can track the subspace for stationary signals, however suffers from a high error floor. This work neutralizes these two issues by proposing an adaptive estimation method, i.e., the P-Oja method. The P-Oja method combines the best features of both algorithms by using small batches and accelerating the Oja's learning rule with a nested power method. Hence, it works well for non-stationary signals and provides a better estimation NOV compared to conventional Oja's learning rule. Our numerical results verify the superiority of the proposed method. We will provide a rigorous analysis for the P-Oja method in our future work.

## 6. REFERENCES

- [1] 5GPP. View on 5G architecture. [Online]. Available: <https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-5G-Architecture-WP-July-2016.pdf>
- [2] E. Axell, G. Leus, E. G. Larsson, and H. V. Poor, "Spectrum sensing for cognitive radio: State-of-the-art and recent advances," *IEEE Signal Process. Mag.*, vol. 29, no. 3, pp. 101–116, 2012.
- [3] X. Fu, N. D. Sidiropoulos, J. H. Tranter, and W.-K. Ma, "A factor analysis framework for power spectra separation and multiple emitter localization," *IEEE Trans. on Signal Process.*, vol. 63, no. 24, pp. 6581–6594, December 2015.
- [4] G. H. Golub and C. F. van Loan, *Matrix computations*, 3rd ed. Baltimore, MD: Johns Hopkins University Press, 1996.
- [5] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *Journal of Math. Analysis and Applications*, no. 106, pp. 69–84, 1985.
- [6] L. Li, A. Scaglione, and J. H. Manton, "Distributed principal subspace estimation in wireless sensor networks," *IEEE Journal of Sel. Topics in Signal Process.*, vol. 5, no. 4, pp. 725–738, Aug 2011.
- [7] A. Scaglione, R. Pagliari, and H. Krim, "The decentralized estimation of the sample covariance," in *Proc. Asilomar*, November 2008, pp. 1722–1726.
- [8] 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, May 2016.
- [9] R. Arora, A. Cotter, K. Livescu, and N. Srebro, "Stochastic optimization for PCA and PLS," in *Allerton*, 2012.
- [10] A. Balsubramani, S. Dasgupta, and Y. Freund, "The fast convergence of incremental PCA," in *NIPS*, 2013.
- [11] C.-L. Li, H.-T. Lin, and C.-J. Lu, "Rivalry of two families of algorithms for memory-restricted streaming PCA," in *AISTATS*, 2016.
- [12] O. Shamir, "A stochastic PCA and SVD algorithm with an exponential convergence rate," in *ICML*, 2015.
- [13] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proc. IEEE*, vol. 98, no. 11, pp. 1847–1864, Nov. 2010.
- [14] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems & Control Letters*, no. 53, pp. 65–78, Feb 2004.