Decentralized Eigendecomposition for Online Learning over Graphs

Yufan Fan, Minh Trinh-Hoang, and Marius Pesavento
Communication Systems Group, Technische Universität Darmstadt, Germany

***Abstract*—We address the problem of decentralized eigenvalue decomposition of a general symmetric matrix that is important, e.g., in Principal Component Analysis. The proposed algorithm only uses local interactions among neighboring agents and is based on the representation of the matrix as a recursive update of rank-one components. This makes the algorithm attractive for online eigenvalue and eigenvector tracking applications. We study the performance of the proposed algorithm in two important application examples: First, we consider the online eigendecomposition of a sample covariance matrix over the network. Then, we investigate the online computation of the spectra of the graph Laplacian that is important in, e.g., graph Fourier analysis and graph-dependent filter design. Simulation results reveal that the proposed algorithm outperforms existing decentralized algorithms both in terms of estimation accuracy as well as communication cost.**

I. Introduction

The eigenvalue decomposition is fundamental in various application areas such as signal processing, data mining, and machine learning [1]–[3]. In particular, when the data dimension is large, dimensionality reduction techniques, such as the Principal Component Analysis (PCA), are required to obtain lower-dimensional representations of the data, e.g., by means of eigendecomposition [4]–[6]. In big data applications and when the data is massively distributed over a network of agents, decentralized algorithms are required as scalable solutions. Based on the concept of in-network processing, agents perform local processing and collaborate by exchanging information only within their local neighborhood. A distributed PCA algorithm is proposed in [6], where the local PCA is performed on the local data and then merged at a central coordinator. To employ the PCA fully distributively, consensus gossiping strategies are applied in many decentralized signal processing algorithms [7], [8]. Combined with the Average Consensus (AC) algorithm, the decentralized power method is proposed in [9]. The distributed Oja's method is proposed in [10], where the Oja's rule is performed in a distributed manner. A distributed adaptive algorithm named DACMEE is proposed in [11], where no nested AC iterations are required. However, the applications of this algorithm are limited to fully connected or tree network topologies.

Based on the distributed power method for eigenvalue decomposition, the distributed ESPRIT (d-ESPRIT) algorithm is proposed in [12] for distributed Direction-of-Arrival (DoA) estimation. To reduce the communication cost, the d-ESPRIT is further developed as the L-ESPRIT in [13], where the decentralized Lanczos method is used instead of the distributed power method. Combining the AC algorithm and the non-Hermitian generalized eigendecomposition, an online adaptive algorithm is proposed in [14] to perform the decentralized cooperative DoA tracking.

In this work, we propose a decentralized online adaptive eigendecomposition algorithm that is based on the eigendecomposition of a rank-one modified diagonal matrix [15]. The data available at each agent is diffused through the network using parallel consensus protocols with local interactions between agents. At termination, each agent knows all eigenvalues and the respective rows of the eigenvector matrix that correspond to its own data. The benefit of the proposed decentralized algorithm with respect to the popular decentralized power method [9] is that all eigenvalues and eigenvectors are computed in parallel and that the algorithm is particularly suitable for online tracking applications where rank-one updates are natural. We evaluate the performance of the proposed decentralized eigendecomposition algorithm in two prominent application examples: (A) the decentralized eigendecomposition of the sample covariance matrix and (B) the decentralized online computation of the graph eigenvectors and eigenvalues in a dynamically evolving graphical network.

II. Online Distributed Strategy for Eigenvalue Decomposition

Consider a network of N agents that is described by the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \{1, \ldots, N\}$ is the set of nodes (agents) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges. Assume that the network $\mathcal{G}$ is unweighted, i.e., the graph $\mathcal{G}$ is characterized by its symmetric adjacency matrix $\mathbf{A} = [a_{ij}] \in \mathbb{R}^{N \times N}$. The entry a_{ij} is 1 if $(i, j) \in \mathcal{E}$, i.e., if agent i has a communication link to agent j, and 0 otherwise. Let $d_i = \sum_{j=1}^{N} a_{ij}$ denote the degree of node i, then $\mathbf{D} = \operatorname{diag}(d_1, d_2, \ldots, d_N)$ is the degree matrix of $\mathcal{G}$. The corresponding graph Laplacian matrix can be expressed as $\mathbf{L} = \mathbf{D} - \mathbf{A}$. Let $x_i(t) \in \mathbb{R}$ denote the signal of node i at the time instant t, and the vector $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_N(t)]^\intercal \in \mathbb{R}^N$ collects the signal of all nodes in the network.

We address the problem of the online distributed computation of the eigenvalues of a rank-one modification

$$\mathbf{R}(t) = \mathbf{R}(t-1) + \rho(t)\mathbf{x}(t)\mathbf{x}(t)^\intercal. \tag{1}$$

where $\rho(t) \in \{-1, 1\}$. We assume that the eigenvalues $\mathbf{\Lambda}(t-1) = \operatorname{diag}(\lambda_1(t-1), \ldots, \lambda_N(t-1))$ and corresponding eigenvectors $\mathbf{U}(t-1) = [\mathbf{u}_1(t-1), \ldots, \mathbf{u}_N(t-1)]$ of $\mathbf{R}(t-1)$ are known, which are related as follows

$$\mathbf{U}(t-1)^\intercal \mathbf{R}(t-1)\mathbf{U}(t-1) = \mathbf{\Lambda}(t-1). \tag{2}$$

Furthermore, we assume the eigenvalues are distinct and sorted in descending order as $\lambda_1(t-1) > \cdots > \lambda_N(t-1)$.

Multiplying both sides of (1) with $\mathbf{U}(t-1)^\intercal$ and $\mathbf{U}(t-1)$ from the left and the right, respectively, leads to

$$\mathbf{U}(t-1)^\intercal \mathbf{R}(t)\mathbf{U}(t-1) = \mathbf{\Lambda}(t-1) + \rho(t)\mathbf{z}(t)\mathbf{z}(t)^\intercal, \quad (3)$$

with

$$\mathbf{z}(t) = [z_1(t), \ldots, z_N(t)]^\intercal = \mathbf{U}(t-1)^\intercal \mathbf{x}(t). \quad (4)$$

The expression on the right hand side of (3) represents a rank-one modification of a diagonal matrix. The modified eigenvalues and corresponding modified eigenvectors are denoted as $\bar{\mathbf{\Lambda}}(t-1) = \operatorname{diag}\left(\bar{\lambda}_1(t-1), \ldots, \bar{\lambda}_N(t-1)\right)$ and $\mathbf{V}(t-1) = [\mathbf{v}_1(t-1), \ldots, \mathbf{v}_N(t-1)]$, respectively, which are related as

$$\begin{aligned} &\mathbf{V}(t-1)^\intercal(\mathbf{\Lambda}(t-1) + \rho(t)\mathbf{z}(t)\mathbf{z}(t)^\intercal)\mathbf{V}(t-1) = \bar{\mathbf{\Lambda}}(t-1), \\ &\mathbf{V}(t-1)^\intercal \mathbf{U}(t-1)^\intercal \mathbf{R}(t)\mathbf{U}(t-1)\mathbf{V}(t-1) = \bar{\mathbf{\Lambda}}(t-1). \end{aligned} \quad (5)$$

Furthermore, since

$$\mathbf{U}(t)^\intercal \mathbf{R}(t)\mathbf{U}(t) = \mathbf{\Lambda}(t), \quad (6)$$

we observe that $\mathbf{R}(t)$ shares the same eigenvalues with the rank one modified matrix, i.e., $\mathbf{\Lambda}(t) = \bar{\mathbf{\Lambda}}(t-1)$, and the corresponding eigenvectors can be computed by

$$\mathbf{U}(t) = \mathbf{U}(t-1)\mathbf{V}(t-1). \quad (7)$$

A. Rational Function Approximation Approach

As described above, the eigenvalue decomposition of a diagonal matrix modified by a rank-one matrix in (5) plays a crucial role in our proposed distributed algorithm. For notational simplicity, we drop the dependence of the matrix arguments on the time instant t. By exploiting the structure of the matrix argument, the efficient implementation of the rank-one modification problem is facilitated by the following theorem [16]:

Theorem 1: Suppose $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_N) \in \mathbb{R}^{N\times N}$ where the diagonal entries are distinct and are sorted in descending order, i.e., $\lambda_1 > \cdots > \lambda_N$. Further assume that $\rho \neq 0$ and $\mathbf{z} = [z_1, \ldots, z_N] \in \mathbb{R}^N$ with $z_i \neq 0$ for all $i = 1, \ldots, N$. If $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_N] \in \mathbb{R}^{N\times N}$ is an orthogonal matrix such that

$$\mathbf{V}^\intercal(\mathbf{\Lambda} + \rho\mathbf{z}\mathbf{z}^\intercal)\mathbf{V} = \operatorname{diag}\left(\bar{\lambda}_1, \ldots, \bar{\lambda}_N\right),$$

with $\bar{\lambda}_1 > \cdots > \bar{\lambda}_N$, then

1) The values in set $\{\bar{\lambda}_i\}_{i=1}^N$ are the N zeros of the secular function $f(\lambda) = 1 + \rho\mathbf{z}^\intercal(\mathbf{\Lambda} - \lambda\mathbf{I})^{-1}\mathbf{z}$.
2) The values $\{\bar{\lambda}_i\}_{i=1}^N$ satisfy the interlacing property, i.e., $\bar{\lambda}_1 > \lambda_1 > \bar{\lambda}_2 > \cdots > \bar{\lambda}_N > \lambda_N$, if $\rho > 0$, $\lambda_1 > \bar{\lambda}_1 > \lambda_2 > \cdots > \lambda_N > \bar{\lambda}_N$, if $\rho < 0$.
3) The eigenvector $\mathbf{v}_i$ associated with $\bar{\lambda}_i$ is a multiple of $(\mathbf{\Lambda} - \bar{\lambda}_i\mathbf{I})^{-1}\mathbf{z}$.

There is no loss in generality in assuming that $\rho > 0$, otherwise, we can replace λ_i by $-\lambda_{N-i+1}$ and ρ by $-\rho$ [16]. The eigenvalues $\bar{\lambda}_k$ of the matrix $\mathbf{\Lambda} + \rho\mathbf{z}\mathbf{z}^\intercal$ can be computed by solving $f(\lambda) = 0$, i.e.,

$$f(\lambda) = 1 + \rho\sum_{n=1}^{N}\frac{z_i^2}{\lambda_i - \lambda} = 0. \quad (8)$$

Based on the interlacing property of the eigenvalues, for the k-th eigenvalue $\bar{\lambda}_k \in (\lambda_k, \lambda_{k-1})$ with $\lambda_0 = \lambda_1 + \rho\mathbf{z}^\intercal\mathbf{z}$ [17], we can rearrange the equation as

$$-\psi_{k-1}(\lambda) = 1 + \phi_k(\lambda) \quad (9)$$

where

$$\psi_{k-1}(\lambda) = \rho\sum_{i=1}^{k-1}\frac{z_i^2}{\lambda_i - \lambda} \quad \text{and} \quad \phi_k(\lambda) = \rho\sum_{i=k}^{N}\frac{z_i^2}{\lambda_i - \lambda}. \quad (10)$$

Since both functions $\psi_{k-1}(\lambda)$ and $\phi_k(\lambda)$ are sums of rational functions, it is natural to approximate them with simple rational functions [16] as

$$\tilde{\psi}_{k-1}(\lambda) = p + \frac{q}{\lambda_{k-1} - \lambda} \quad \text{and} \quad \tilde{\phi}_k(\lambda) = r + \frac{s}{\lambda_k - \lambda}. \quad (11)$$

In (11), the parameters p, q, r and s are chosen such that, at the given iteration point $\lambda^{(\tau)}$, the rational approximants $\tilde{\psi}_{k-1}(\lambda)$ and $\tilde{\phi}_k(\lambda)$ in (11) coincide with the true rational functions $\psi_k(\lambda)$ and $\psi_{k-1}(\lambda)$ in (10) up to the first derivative, respectively. The next iteration point $\lambda^{(\tau+1)} \in (\lambda_k, \lambda_{k-1})$ is obtained from a solution of the following equation

$$-\tilde{\psi}_{k-1}(\lambda) = 1 + \tilde{\phi}_k(\lambda). \quad (12)$$

For special case where $k = 1$, $\psi_{k-1}(\lambda)$ is approximated as $\tilde{\psi}_0(\lambda) = 0$. The rational function approximation algorithm is summarized in Algorithm 1 [18] as follows:

Algorithm 1 Computing the k-th Eigenvalue of Rank-One Modification With Rational Function Approximation, $\mathtt{RA}_k(\cdot)$

Initialization: Iteration index $\tau = 0$, scalar ρ, vector $\mathbf{z}$, tolerance ϵ, arbitrary starting point $\lambda^{(\tau)} \in (\lambda_k, \lambda_{k-1})$
repeat
  Find the parameters p and q such that

$$\tilde{\psi}_{k-1}(\lambda^{(\tau)}) = \psi_{k-1}(\lambda^{(\tau)}) \text{ and } \tilde{\psi}'_{k-1}(\lambda^{(\tau)}) = \psi'_{k-1}(\lambda^{(\tau)}). \quad (13)$$

  Find the parameters r and s such that

$$\tilde{\phi}_k(\lambda^{(\tau)}) = \phi_k(\lambda^{(\tau)}) \text{ and } \tilde{\phi}'_k(\lambda^{(\tau)}) = \phi'_k(\lambda^{(\tau)}). \quad (14)$$

  Find $\lambda^{(\tau+1)} \in (\lambda_k, \lambda_{k-1})$ which satisfies

$$-\tilde{\psi}_{k-1}(\lambda^{(\tau+1)}) = 1 + \tilde{\phi}_k(\lambda^{(\tau+1)}) \quad (15)$$

  $\tau \leftarrow \tau + 1$
until $|\lambda^{(\tau+1)} - \lambda^{(\tau)}| < \epsilon$
return $\bar{\lambda}_k = \lambda^{(\tau+1)}$, $\mathbf{v}_k = (\mathbf{\Lambda} - \bar{\lambda}_k\mathbf{I})^{-1}\mathbf{z}/|(\mathbf{\Lambda} - \bar{\lambda}_k\mathbf{I})^{-1}\mathbf{z}|$

B. Push-Sum Consensus

To compute the eigenvalues and eigenvectors over the network as presented in Algorithm 1, the sample vector update $\mathbf{z}(t)$ defined in (4) must be available at all nodes. Nevertheless, each node in the network knows only its corresponding entry in the vector $\mathbf{x}(t)$ locally. Therefore, in this section, we introduce a distributed consensus procedure to compute the update vector $\mathbf{z}(t)$ over the network. We assume that each node maintains at each time instant one row of the eigenvector matrix $\mathbf{U}(t-1)$ and one entry of the signal vector $\mathbf{x}(t)$. The k-th entry of $\mathbf{z}(t)$ for all $k = 1, \ldots, N$ is computed as

$$z_k(t) = \mathbf{e}_k^\intercal \mathbf{U}(t-1)\mathbf{x}(t) = \sum_{i=1}^{N} u_{ik}(t-1)x_i(t), \quad (16)$$

where $\mathbf{e}_k$ is the k-th column of an identity matrix, and $x_i(t)$ and $u_{ik}(t-1)$ are available locally at the i-th node. We remark that the expression in (16) can be computed distributively with different methods, e.g., the averaging consensus algorithm, linear graph filters, and nonlinear graph filters [19], [20]. One candidate for computing (16) distributively is the Push-Sum consensus algorithm, which was first introduced and analyzed in [21], and the convergence of the generalized weighted gossiping algorithm is proven in [22] for any graphs based on weak ergodicity arguments. Benefiting from the drop of the double stochasticity of the updating matrix, the Push-Sum algorithm is naturally applicable to directed networks. Its principle is provided in the following.

Assume that the vector $\mathbf{y} = [y_1, \ldots, y_N]^\intercal$ contains the values of N nodes whose average needs to be computed distributively over the network. From the adjacency matrix of the network, we introduce a column stochastic matrix $\mathbf{P}$ which satisfies $\mathbf{1}^\intercal\mathbf{P} = \mathbf{1}^\intercal$, where $p_{ji} = 0$ if there is no direct edge between node i and j. A simple and sufficient example of matrix $\mathbf{P}$ is

$$p_{ji} = \begin{cases} 1/d_i, & (i,j) \in \mathcal{V}, \\ 0, & \text{otherwise}. \end{cases} \tag{17}$$

In order to perform the averaging operation distributively, the Push-Sum consensus algorithm further assumes that, at a given consensus instant γ, each node i maintains a set consisting of two values: a cumulative estimate of the sum $s_{i(\gamma)}$ and a weight $w_{i(\gamma)}$ for $i = 1, \ldots, N$. The vector of cumulative sums $\mathbf{s}_{(\gamma)} = \left[s_{1(\gamma)}, \ldots, s_{N(\gamma)}\right]^\intercal \in \mathbb{R}^N$ and the vector of weights $\mathbf{w} = \left[w_{1(\gamma)}, \ldots, w_{N(\gamma)}\right]^\intercal \in \mathbb{R}^N$ are initialized as

$$\mathbf{s}_{(0)} = \mathbf{y} \text{ and } \mathbf{w}_{(0)} = \mathbf{1}, \tag{18}$$

respectively. The Push-Sum consensus algorithms consists of two steps, which are iteratively performed in all nodes of the network until convergence. At the γ-th consensus instant, based on the chosen column stochastic matrix $\mathbf{P}$, node i first splits its total sum $s_{i(\gamma)}$ and weight $w_{i(\gamma)}$ into shares and sends to its neighboring node j the corresponding share $\mathbb{S}_{i\to j(\gamma)} = \left\{p_{ji}s_{i(\gamma)}, p_{ji}w_{i(\gamma)}\right\}$. Then, each node updates its own sum and weight by summing up all the shares received from its adjacent nodes. The above mentioned process are summarized in vector form as

$$\mathbf{s}_{(\gamma)} = \mathbf{P}\mathbf{s}_{(\gamma-1)} \text{ and } \mathbf{w}_{(\gamma)} = \mathbf{P}\mathbf{w}_{(\gamma-1)}. \tag{19}$$

Given the estimated sum and weight in (19), the estimated average is calculated at each node by

$$\hat{\mathbf{z}}_{(\gamma)} = \mathbf{s}_{(\gamma)} \oslash \mathbf{w}_{(\gamma)}, \tag{20}$$

where $\oslash$ is Hadamard division. At the end of the Push-Sum iteration, each node shares the same average value, i.e.,

$$\lim_{\gamma\to\infty} \hat{z}_{i(\gamma)} = \lim_{\gamma\to\infty} \frac{s_{i(\gamma)}}{w_{i(\gamma)}} = \frac{\mathbf{1}^\intercal\mathbf{y}}{N}, \text{ for all } \ i \in \mathcal{V}. \tag{21}$$

Remark: If only one node starts with weight 1, then the value computed at the nodes converges to the sum $\mathbf{1}^\intercal\mathbf{y}$, instead of their average. Furthermore, if each node starts with value $s_{i(\gamma)} = 1$, the network size can be determined distributively.

Now the computation of (16) can be achieved distributively, and (4) can be performed by N parallel consensus steps. Moreover, since each node has the access to one row of $\mathbf{U}(t-1)$ and full knowledge of $\mathbf{V}(t-1)$, the update (7) for each row can be performed locally at each node easily with

$$\mathbf{e}_i^\intercal\mathbf{U}(t) = \mathbf{e}_i^\intercal\mathbf{U}(t-1)\mathbf{V}(t-1), \text{ for all } i \in \mathcal{V}. \tag{22}$$

The Push-Sum consensus protocol can be summarized in Algorithm 2 as follows:

Algorithm 2 Push-Sum Consensus Protocol Performed at the i-th Node for the k-th Entry of $\mathbf{z}(t)$, $\mathtt{PS}_k(\cdot)$

Initialization: Iteration index $\gamma = 0$, maximum number of consensus iterations Γ, p_{ji}, initialize $s_{i(0)} = u_{ik}(t)x_i(t)$ and $w_{i(0)} = 1$.
while $\gamma \leq \Gamma$ **do**
  Push step: Send the shares $\mathbb{S}_{i\to j(\gamma)}$ to all adjacent nodes $j \in \mathcal{V}$ with $(i,j) \in \mathcal{E}$.
  Sum step: Sum the shares $\mathbb{S}_{j\to i(\gamma)}$ obtained from all adjacent nodes $j \in \mathcal{V}$ with $(j,i) \in \mathcal{E}$.
  $\gamma \leftarrow \gamma + 1$
end while
return $z_k(t) = Ns_{i(\gamma+1)}/w_{i(\gamma+1)}$

C. Online Distributed Eigenvalue Decomposition Protocol

After applying the Push-Sum consensus protocol to diffuse the information of new sample vector $\mathbf{x}(t)$, and applying rational function approximation to compute the eigenvalues of the rank-one modification, we are able to track the eigenvalues of $\mathbf{R}(t)$ in an online manner for new samples $\mathbf{x}(t)$ that are collected distributively over the network. Our proposed algorithm to track the eigenvalues of the matrix with the rank-one modification (1) is summarized in Algorithm 3 as follows:

Algorithm 3 Online Distributed Eigenvalue Computation Performed at Node i

Initialization: $\mathbf{\Lambda}(0) = \mathbf{0}, \mathbf{e}_i^\intercal\mathbf{U}(0) = \mathbf{e}_i^\intercal$, $\rho(t)$, $t = 1$
while $\mathbf{x}(t)$ observed **do**
  Network Communication (Push-Sum)
  for all $k \in [1, \ldots, N]$ **do in parallel**
    $z_k(t) = \mathtt{PS}_k(\mathbf{e}_i^\intercal\mathbf{U}(t-1)\mathbf{x}(t))$
  end for
  Node Computation (local update)
  for all $k \in [1, \ldots, N]$ **do in parallel**
    $[\bar{\lambda}_k(t-1), \mathbf{v}_k(t-1)] = \mathtt{RA}_k(\mathbf{\Lambda}(t-1) + \rho\mathbf{z}(t)\mathbf{z}(t)^\intercal)$
  end for
  Update $\mathbf{\Lambda}(t) = \bar{\mathbf{\Lambda}}(t-1)$
  Update $\mathbf{e}_i^\intercal\mathbf{U}(t) = \mathbf{e}_i^\intercal\mathbf{U}(t-1)\mathbf{V}(t-1)$ (Eq. (22))
end while

At each time instant t, the i-th node contributes a new sample $x_i(t)$ and maintains a row of $\mathbf{U}(t)$. By running the Push-Sum consensus protocol, an instant of the vector $\mathbf{z}(t)$ is computed in each node throughout the network. Then the rational function approximation is performed locally in each node since $\mathbf{\Lambda}(t), \rho(t)$, and $\mathbf{z}(t)$ are accessible to all the nodes. Depending on the computation and storage capacity, each node can perform the rational function approximation fully parallelized or sequentially.

Remark: In the first $N-1$ samples with $t = 1, \ldots, N-1$, the sample covariance matrix $\mathbf{R}(t)$ has zero eigenvalues with multiplicity larger than one, which violates the assumptions in *Theorem 1*. Thus, an extra deflation step is required to remove the multiplicity to have a rank-one modification with a smaller size. The deflation technique is discussed in [16] and the potential case where $\mathbf{z}(t)$ contains zero components can also be deflated.

III. Case Study

To evaluate the performance of our proposed algorithm, we studied two application cases, and the relative errors are defined as

$$\eta_k(t) = \frac{|\lambda_k(t) - \hat{\lambda}_k(t)|}{|\hat{\lambda}_k(t)|}, \quad \tilde{\eta}_k(t) = \frac{|\tilde{\lambda}_k(t) - \hat{\lambda}_k(t)|}{|\hat{\lambda}_k(t)|}, \tag{23}$$

where $\lambda_k(t)$ is the k-th eigenvalue computed by our proposed algorithm, $\tilde{\lambda}_k(t)$ is the k-th eigenvalue computed by a centralized processor, and $\hat{\lambda}_k$ is the k-th eigenvalue of the true sample distribution. Throughout this section, the numerical precision for the rational function approximation is set as $\epsilon = 10^{-12}$.

A. Application Example 1: Eigenvalue Decomposition of Sample Covariance Matrix

One natural application example with our proposed algorithm is the computation of the eigenvalues of the sample covariance matrix. In this case, $x_i(t)$ is the observation obtained at the i-th node, and $\mathbf{R}(t) = \sum_{m=0}^{t} \mathbf{x}(m)\mathbf{x}(m)^\intercal$ is the sample covariance matrix. A decentralized power method proposed in [9] is used as a comparison, where the number of power method iterations is denoted as Ω.

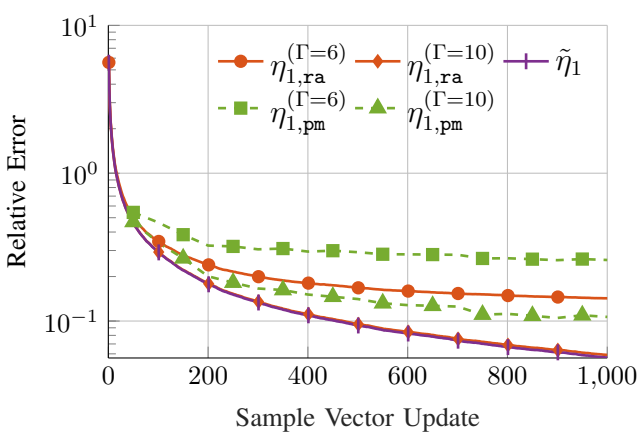

Fig. 1. Relative error of λ_1 for $\mathbf{R}(t)$ with different maximum number of consensus iterations Γ, where $\eta_{\mathtt{ra}}$ and $\eta_{\mathtt{pm}}$ stand for relative errors using our proposed online algorithm with rational function approximation and using distributed power method ($\Omega = 30$), respectively.

In Figure 1, we observe that more Push-Sum or Average Consensus iterations are required for both protocols to have a better error performance compared to the centralized algorithm. The distributed algorithms can compute the exact eigenvalues when they run infinite consensus gossiping.

Although the relative error associated with the distributed power method is comparable to that of our proposed algorithm, the total number of consensus rounds of the power method is higher than that of our proposed method. In order to measure the amount of required data exchange for the consensus-based algorithms, we consider the total communication cost denoted as C. The communication cost C is defined as the number of total consensus rounds that have been performed to compute all the eigenvalues, where one Push-Sum round is treated as two Average Consensus rounds since two values are maintained in each Push-Sum iteration. More precisely, for an undirected network with N nodes and T sample vectors, our proposed algorithm requires $C_{\mathtt{ra}} = 2NT$ consensus rounds for the computation of all eigenvalues, while the distributed power method requires a higher number of total consensus rounds, i.e., $C_{\mathtt{pm}} = N(T\Omega + T + 2) + \Omega N(N-1)/2$, including distributed normalization and largest eigenvalue subtraction.

Furthermore, the distributed power method requires the knowledge of all sample vectors to perform the eigenvalue decomposition, where our proposed algorithm is an online algorithm that can update the eigenvalue each time when new sample vector is obtained. Note that the power method naturally requires more iterations to achieve the stationary point if the largest eigenvalue is not dominant over all eigenvalues, which our proposed algorithm does not suffer from.

B. Application Example 2: Spectrum Computation in Dynamic Graphs.

In this application example, we consider distributed online estimation of the spectrum of the Graph Laplacian for the columns of the oriented incidence matrix $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_{N_e}] \in \mathbb{R}^{N \times N_e}$, where $N_e = |\mathcal{E}|$. The element $b_{k\ell}$ for node k and edge ℓ (connecting node i and j) is given by

$$b_{k\ell} = \begin{cases} 1, & \text{if } k = i,\ \ell = j \\ -1, & \text{if } k = j,\ \ell = i \\ 0, & \text{otherwise.} \end{cases} \tag{24}$$

The graph Laplacian matrix can be expressed as

$$\mathbf{L} = \mathbf{B}\mathbf{B}^\intercal = \sum_{\ell=1}^{N_e} \mathbf{b}_\ell \mathbf{b}_\ell^\intercal. \tag{25}$$

Assuming that the vertices in the network are labeled, an appropriate protocol can be designed, where the agents cooperatively update the graph Laplacian with rank-one modifications according to (25). The eigendecomposition can then be computed using Algorithm 3. This facilitates the distributed computation of the Graph Fourier transform and distributed graph-dependent graph filter design. The relative error performance and communication cost of our proposed algorithm and the distributed power method are shown in Figure 2.

One advantage of our proposed method compared to the distributed power method is the ability to efficiently track eigenvalues of Laplacian matrices associated with dynamically evolving graphs. For the sake of simplicity, we assume a given graph Laplacian $\mathbf{L}(t-1)$ at instant $t-1$. Further assume that at instant t, a random edge ℓ disappears in the network. Then we can use equation (1) with $\mathbf{R}(t-1) = \mathbf{L}(t-1)$, $\mathbf{x}(t) = \mathbf{b}_\ell$, and $\rho(t) = -1$ to express the Laplacian at instant t as a rank-one update. Similarly, if at instant t a new edge n appears in the

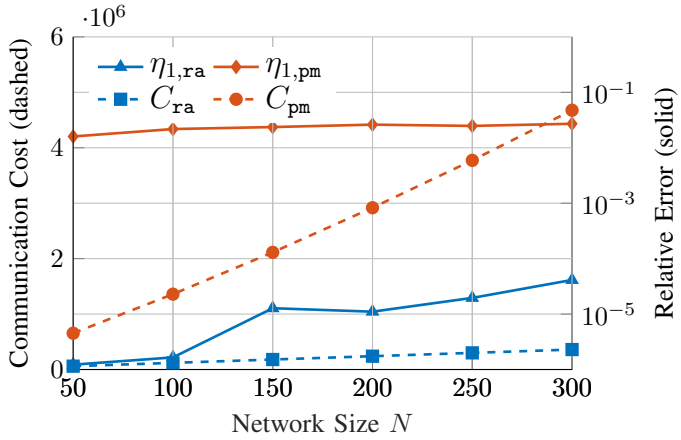

Fig. 2. Relative error of λ_1 and total communication cost C of all eigenvalues for graph Laplacian matrix of d-regular ($d = 4$) networks with different network size N, where $\Gamma = 100$ and $\Omega = 20$.

network we choose $\rho(t) = 1$ in the update. Figure 3 shows the largest eigenvalue tracking when the network evolves, where the network is initialized as an undirected d-regular network with $d = 4$, and $N = 50$ nodes. For the first 100 iterations, our algorithm evaluates over all existing edges and computes the eigenvalue. Then, the network starts evolving, i.e., edges are randomly removed or added, and the relative error behavior shows that our algorithm is able to track the evolution of the network in the eigenvalues.

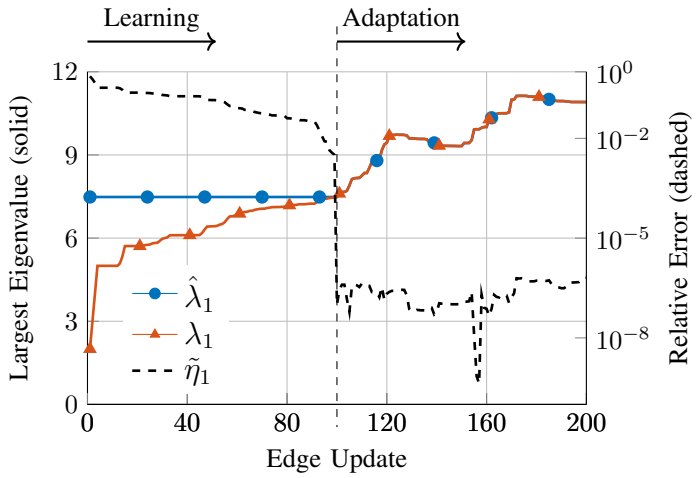

Fig. 3. Eigenvalue learning and adaptation for dynamic graph network with $N = 50$ nodes and $\Gamma = 100$.

IV. Conclusion

In this paper we propose a decentralized online eigendecomposition algorithm for parallel tracking of all eigenvalues of a rank-one modified matrix. Our proposed algorithm is based on parallel averaging consensus steps and local rational function approximations. Our decentralized solution of the proposed algorithm converges to the centralized solution at a reduced total communication cost compared to the distributed power method.

References

[1] S. Mostafa, L. Tang, and F. Wu, "Diagnosis of Autism Spectrum Disorder Based on Eigenvalues of Brain Networks," *IEEE Access*, vol. 7, pp. 128 474–128 486, 2019.

[2] P. Urriza, E. Rebeiz, and D. Cabric, "Eigenvalue-Based Cyclostationary Spectrum Sensing Using Multiple Antennas," in *IEEE Global Communications Conference (GLOBECOM)*, 2012, pp. 1501–1506.

[3] R. Gui, X. Xu, D. Zhang, L. Wang, R. Yang, and F. Pu, "Built-Up Areas Extraction from Polsar Imagery Via Eigenvalue Statistical Information and Pu-Learning," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2019, pp. 1196–1199.

[4] K. Pearson, "On Lines and Planes of Closest Fit to Systems of Points in Space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[5] H. Hotelling, "Analysis of a Complex of Statistical Variables Into Principal Components," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[6] Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist, "Principal Component Analysis for Dimension Reduction in Massive Distributed Data Sets," in *Proceedings of IEEE International Conference on Data Mining (ICDM)*, vol. 1318, no. 1784, 2002, p. 1788.

[7] L. Xiao, S. Boyd, and S. Lall, "A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus," in *4th International Symosium on Information Processing in Sensor Networks.*, 2005, pp. 63–70.

[8] R. Olfati-Saber, "Distributed Kalman Filter With Embedded Consensus Filters," in *Proceedings of the 44th IEEE Conference on Decision and Control*, 2005, pp. 8179–8184.

[9] A. Scaglione, R. Pagliari, and H. Krim, "The Decentralized Estimation of the Sample Covariance," in *42nd Asilomar Conference on Signals, Systems and Computers*, 2008, pp. 1722–1726.

[10] L. Li, A. Scaglione, and J. H. Manton, "Distributed Principal Subspace Estimation in Wireless Sensor Networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 725–738, 2011.

[11] A. Bertrand and M. Moonen, "Distributed Adaptive Estimation of Covariance Matrix Eigenvectors in Wireless Sensor Networks With Application to Distributed PCA," *Signal Processing*, vol. 104, pp. 120–135, 2014.

[12] W. Suleiman, M. Pesavento, and A. M. Zoubir, "Performance Analysis of the Decentralized Eigendecomposition and ESPRIT Algorithm," *IEEE Transactions on Signal Processing*, vol. 64, no. 9, pp. 2375–2386, 2016.

[13] W. Suleiman, P. Parvazi, M. Pesavento, and A. Zoubir, "Decentralized Direction Finding Using Lanczos Method," in *IEEE 8th Sensor Array and Multichannel Signal Processing Workshop (SAM)*, 2014, pp. 9–12.

[14] W. Suleiman, M. Pesavento, and A. M. Zoubir, "Decentralized Cooperative DOA Tracking Using Non-Hermitian Generalized Eigendecomposition," in *23rd European Signal Processing Conference (EUSIPCO)*, 2015, pp. 2626–2630.

[15] G. H. Golub, "Some Modified Eigenvalue Problems," *SIAM Review*, vol. 15, no. 2, pp. 318–334, 1973.

[16] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-One Modification of the Symmetric Eigenproblem," *Numerische Mathematik*, vol. 31, no. 1, pp. 31–48, 1978.

[17] R.-C. Li, "Solving Secular Equations Stably and Efficiently," EECS Department, University of California, Berkeley, Tech. Rep. UCB/CSD-94-851, Dec 1994.

[18] M. Trinh-Hoang, M. Viberg, and M. Pesavento, "Partial Relaxation Approach: An Eigenvalue-Based DOA Estimator Framework," *IEEE Transactions on Signal Processing*, vol. 66, no. 23, pp. 6190–6203, Dec. 2018.

[19] M. Contino, E. Isufi, and G. Leus, "Distributed Edge-Variant Graph Filters," in *IEEE 7th Int. Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2017, pp. 1–5.

[20] B. Iancu and E. Isufi, "Towards Finite-Time Consensus With Graph Convolutional Neural Networks," in *2020 28th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 2145–2149.

[21] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-Based Computation of Aggregate Information," in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, 2003, pp. 482–491.

[22] F. Bénézit, V. Blondel, P. Thiran, J. Tsitsiklis, and M. Vetterli, "Weighted Gossip: Distributed Averaging Using Non-Doubly Stochastic Matrices," in *IEEE International Symposium on Information Theory*, 2010, pp. 1753–1757.