Accelerated Algorithms for Eigen-Value Decomposition with Application to Spectral Clustering

Songtao Lu and Zhengdao Wang
Department of Electrical and Computer Engineering, Iowa State University, Ames, IA 50011, USA
Emails: {songtao, zhengdao}@iastate.edu

***Abstract*—Fast and accurate numerical algorithms for Eigen-Value Decomposition (EVD) are of great importance in solving many engineering problems. In this paper, we aim to develop algorithms for finding the leading eigen pairs with improved convergence speed compared to existing methods. We introduce several accelerated methods based on the power iterations where the main modification is to introduce a memory term in the iteration, similar to Nesterov's acceleration. Results on convergence and the speed of convergence are presented on a proposed method termed Memory-based Accelerated Power with Scaling (MAPS). Nesterov's acceleration for the power iteration is also presented. We discuss possible application of the proposed algorithm to (distributed) clustering problems based on spectral clustering. Simulation results show that the proposed algorithms enjoy faster convergence rates than the power method for matrix eigen-decomposition problems.**

***Index Terms*—Eigen-value decomposition, Nesterov acceleration, spectral clustering, distributed implementation**

I. Introduction

Matrix eigen-value decomposition (EVD) is essential in many engineering problems, and has been studied for many years [1]. As one example, for clustering problems, a well known method termed spectral clustering relies on finding the first several smallest eigenvalues and the corresponding eigenvectors of the Laplacian matrix of a graph, see e.g., a tutorial of spectral clustering [2]. Recently, spectral clustering was used to study social networks in [3], [4] and analyze variability of high-dimensional data sets.

Power method is one of the classical methods, which is simple and can decompose the matrix through parallel computing efficiently [1], [5]. Depending on the distribution of the eigenvalues, it may suffer a slow convergence rate. Another type of methods is based on a matrix transformation, such as Arnoldi's algorithm [6] and Lanczos' method [7] which are an adaptation of power method with fast convergence rate but their implementation complexity and storage requirement are higher than those of the power method. Moreover, these algorithms are not easy to be implemented distributively. For large-scale EVD problems, the convergence rate, accuracy, computational cost, memory requirement, and possibility of parallel and/or distributed implementation, are all important factors in the design of the algorithms.

In this paper, we aim to develop algorithms for the EVD problem with faster convergence rates. We propose three modifications of the power iteration by including a memory term from the previous iteration. The algorithms are inspired by similar memory terms in approximate message passing (AMP) and Nesterov's acceleration. For one algorithm, the so-termed memory-based accelerated-power with scaling (MAPS) algorithm, we also include results on conditions for guaranteeing convergence and maximization of convergence speed through proper parameter selection. We discuss possible application of the proposed algorithms to spectral clustering problems. Numerical simulation results show that the proposed algorithms enjoy faster convergence rate compared with the well-known power method.

II. Power method

Power method is a well-known algorithm for matrix EVD [1]. It is a very simple algorithm that involves the following iteration

$$\begin{aligned} \widetilde{\mathbf{x}}^{(t+1)} &= \mathbf{A}\mathbf{x}^{(t)}, \\ \mathbf{x}^{(t+1)} &= \frac{\widetilde{\mathbf{x}}^{(t+1)}}{\|\widetilde{\mathbf{x}}^{(t+1)}\|_2} \end{aligned} \tag{1}$$

where $\mathbf{A}$ is a given square matrix, $\mathbf{x}^{(t)}$ and $\widetilde{\mathbf{x}}^{(t)}$ are column vectors, and t denotes the index of iterations. It is not necessary that the normalization step is implemented at each step. When the eigenvalue of the largest magnitude is strictly larger than other eigenvalues in magnitude, and it has multiplicity one, the iteration is guaranteed to converge to the leading eigenvector of $\mathbf{A}$. The convergence rate of the power method is linear (or geometric) with ratio

$$\chi_{\text{power}} = \frac{|\lambda_2|}{|\lambda_1|} \tag{2}$$

where λ_1 is the largest eigenvalue and λ_2 is the second largest one both in magnitude. This ratio of the first two dominant eigenvalues of $\mathbf{A}$ can be improved by the shifted power method, which uses $\mathbf{A} - \sigma\mathbf{I}$ instead of $\mathbf{A}$ such that the power method can be accelerated [1], where σ is a suitably chosen scalar.

III. Proposed Algorithms

Based on the AMP algorithm, modified power methods were proposed for solving cone-constraint principle component analysis (PCA) problem [8] and nonnegative PCA problem

[9], where a simple memory term is incorporated in the power method. With this consideration, the algorithms can give an exact distributional characterization under suitable probabilistic models for a matrix $\mathbf{A}$ [9]. Inspired by such methods, we next propose several methods that accelerate the power iterations.

A. Memory-based Accelerated-Power Algorithm

In general the convergence rate of the power method may be slow. When we perform the iterations, e.g., the tth iteration, which is $\mathbf{A}^t\mathbf{x}^{(0)}$, all the historical estimates $\mathbf{A}^{t-1}\mathbf{x}^{(0)}, \ldots, \mathbf{A}\mathbf{x}^{(0)}$ are discarded. The Arnoldi algorithm and the Lanczos algorithm can make use of these discarded information and construct a basis of the Krylov subspace associated with the matrix $\mathbf{A}$ [6], [7].

Inspired by AMP algorithm, the Arnoldi algorithm, and the Lanczos algorithm, an efficient algorithm that makes use of historical information of the vector $\mathbf{x}^{(t)}$ is proposed. We term the method Memory-based Accelerated-Power (MAP) algorithm, where one-step previous update is incorporated. The MAP iteration is as follows

$$\widetilde{\mathbf{x}}^{(t+1)} = \mathbf{A}\mathbf{x}^{(t)} - \rho^{(t)}\widetilde{\mathbf{x}}^{(t-1)}, \tag{3}$$

$$\mathbf{x}^{(t+1)} = \frac{\widetilde{\mathbf{x}}^{(t+1)}}{\|\widetilde{\mathbf{x}}^{(t+1)}\|_2} \tag{4}$$

where $\rho^{(t)}$ is a scalar and needs to be chosen such that the algorithm can converge. The algorithm is basically power iteration with an additional memory term. It performs normalization (4) at each iteration. The MAP algorithm is shown in Algorithm 1, where T is the total number of iterations.

Algorithm 1 MAP Algorithm

1: Initialization: choose the entries of $\mathbf{x}^{(1)}$ from $(-1/\sqrt{N}, 1/\sqrt{N})$ randomly and set $\mathbf{x}^{(0)} = [0, \ldots, 0]^T$.
2: **for** $t = 1 : T$ **do**
3: $\quad\widetilde{\mathbf{x}}^{(t+1)} = \mathbf{A}\mathbf{x}^{(t)} - \rho^{(t)}\widetilde{\mathbf{x}}^{(t-1)}$
4: $\quad\mathbf{x}^{(t+1)} = \mathbf{x}^{(t+1)}/\|\mathbf{x}^{(t+1)}\|_2$
5: **end for**

B. Memory-based Accelerated-Power with Scaling

The convergence behavior of the MAP algorithm depends on the parameters $\rho^{(t)}$. As such it is difficult to analyze its convergence. A variant of it is proposed next, which we term the MAPS algorithm. Notice that in (3) there is a balance between making updates from the power iterations and retaining historical information of $\widetilde{\mathbf{x}}^{(t)}$. Here we replace the normalization step by a simple scaling factor d and the forgetting parameter $\rho^{(t)}$ is set as a constant ρ. The proposed MAPS is shown in Algorithm 2, where the occasional real normalization (every T' iterations) is also introduced, and ν is a small positive number, and $\widehat{\lambda}_1 = |(\mathbf{x}^{(t)})^T\mathbf{A}\mathbf{x}^{(t)}|$, which is an estimate of the magnitude of the leading eigenvalue.

Algorithm 2 MAPS Algorithm

1: Initialization: we choose the entries of $\mathbf{x}^{(1)}$ from $\{-1/\sqrt{N}, 1/\sqrt{N}\}$ randomly and set $\mathbf{x}^{(0)} = [0, \ldots, 0]^T$.
2: **for** $t = 1 : T$ **do**
3: $\quad\mathbf{x}^{(t+1)} = d\mathbf{A}\mathbf{x}^{(t)} - \rho\mathbf{x}^{(t-1)}$
4: $\quad$**if** $\mathrm{mod}(t, T') = 0$ **then**
5: $\quad\quad\mathbf{x}^{(t)} = \mathbf{x}^{(t)}/\|\mathbf{x}^{(t)}\|_2$
6: $\quad\quad d = (\rho + 1 + \nu)/\widehat{\lambda}_1$
7: $\quad$**end if**
8: **end for**
9: **Output**: $\mathbf{x}^{(T)} = \mathbf{x}^{(T)}/\|\mathbf{x}^{(T)}\|_2$

1) Convergence of MAPS: If the eigenvalues of a matrix $\mathbf{A}$ are $\lambda_1, \ldots, \lambda_N$ which are real and $|\lambda_1| > |\lambda_2| > \ldots |\lambda_N|$. We have the following results regarding the convergence behavior of the MAPS algorithm.

Theorem 1: If we choose $d > (\rho + 1)/\lambda_1$, then

$$\begin{aligned} \lim_{t\to\infty} (\mathbf{x}^{(t)})^T\mathbf{A}\mathbf{x}^{(t)} &= |\lambda_1| \\ \lim_{t\to\infty} \mathbf{x}^{(t)} &= \mathbf{x}_1 \end{aligned} \tag{5}$$

where $\mathbf{x}_1$ is the eigenvector corresponding to the leading eigenvalue in-magnitude of the matrix $\mathbf{A}$.

Theorem 2: If we choose ρ such that

$$\left(\frac{|\lambda_1|}{|\lambda_2|} - \sqrt{\frac{\lambda_1^2}{\lambda_2^2} - 1}\right)^2 < \rho < \left(\frac{|\lambda_1|}{|\lambda_2|} + \sqrt{\frac{\lambda_1^2}{\lambda_2^2} - 1}\right)^2, \tag{6}$$

the optimal d of the MAPS algorithm that maximizes the asymptotic convergence ratio is

$$d^* = \frac{2\sqrt{\rho}}{|\lambda_2|}. \tag{7}$$

When the optimal d is chosen, the convergence of the MAPS algorithm d is linear, with ratio

$$\chi_{\mathsf{MAPS}} = \epsilon\frac{|\lambda_2|}{|\lambda_1|} \tag{8}$$

where

$$\epsilon = \frac{1}{1 + \sqrt{\left(1 - \left(\frac{\lambda_2}{\lambda_1}\right)^2\right)}}. \tag{9}$$

Several remarks are in order.

Remark 1. Note that $\rho = 1$ satisfies this condition (6).

Remark 2. Since λ_2 in general is not known before the algorithm starts, we may adopt $\widehat{\lambda}_1$, which is at most λ_1, as an replacement of $|\lambda_2|$ for setting d according to (7).

Remark 3. Because $\epsilon < 1$, the convergence rate of the MAPS algorithm with the optimal d is faster than that of the power iteration.

Due to space limit, we only provide a sketch of the proof of Theorem 1 in the appendix. The proof of Theorem 2 is omitted.

C. Nesterov-Power Method

The Nesterov method [10] also keeps the historical information of the previous iterations. It combines the current estimate and the previous one step estimate as the initialization of the new estimate. In this way, the gradient descent method can be accelerated.

Inspired by the Nesterov method and our proposed MAPS algorithm, we can update the power method as follows,

$$\mathbf{x}^{(t+1)} = \mathbf{A}\mathbf{y}^{(t)}, \tag{10}$$

$$\mathbf{y}^{(t)} = \mathbf{x}^{(t)} + \beta^{(t)}(\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)}) \tag{11}$$

where $\beta^{(t)}$ is the step size which can be chosen based on different protocols.

The modified power iteration based on the Nesterov method, called Nesterov-Power, is shown in Algorithm 3. It is worth noting that we can also use a scalar instead of the normalization for the Nesterov-Power method as in Algorithm 2.

Algorithm 3 Nesterov-Power Method

1: Initialization: we choose the entries of $\mathbf{x}^{(1)}$ from $\{-1/\sqrt{N}, 1/\sqrt{N}\}$ randomly and set $\mathbf{x}^{(0)} = [0, \ldots, 0]^T$.
2: **for** $t = 1 : T$ **do**
3: $\quad \mathbf{y}^{(t)} = \mathbf{x}^{(t)} + \beta^{(t)} \left(\mathbf{x}^{(t)} - \mathbf{x}^{(t-1)}\right)$
4: $\quad \widetilde{\mathbf{x}}^{(t+1)} = \mathbf{A}\mathbf{y}^{(t)}$
5: $\quad \mathbf{x}^{(t+1)} = \widetilde{\mathbf{x}}^{(t+1)} / \|\widetilde{\mathbf{x}}^{(t+1)}\|_2$
6: **end for**

IV. Application to Spectral Clustering

The proposed algorithms are useful for obtaining the first few largest eigen pairs. One application, among many others, is for solving the spectral clustering problem, which is an effective method for clustering nodes in a graph [2]. The method depends on finding the first several smallest eigen pairs of the normalized Laplacian matrix. Equivalently, we are interested in finding several largest eigenvalues and the corresponding eigenvectors of the normalized adjacency matrix. Let $\mathbf{A}$ denote the adjacency matrix of a graph, whose entry $\mathbf{A}_{i,j}$ indicates the connection between the ith node and the jth node. For example, for an unweighted graph, the definition of the adjacency matrix is

$$\mathbf{A}_{i,j} = \begin{cases} 1, & \text{if } i\text{th node and } j\text{th node are connected,} \\ 0, & \text{otherwise.} \end{cases}$$

The normalized adjacency matrix is defined as

$$\widetilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2} \tag{12}$$

where the degree matrix is $\mathbf{D} = \text{diag}\{d_1, \ldots, d_N\}$ and d_i is the total weight of edges connected to node i.

After getting the largest eigenvalue λ_1 and the corresponding eigenvector $\mathbf{x}_1$, we can obtain the second largest eigenvalue and the corresponding eigenvector of $\widetilde{\mathbf{A}}$ by matrix deflation, i.e., finding the largest eigenvalue and the corresponding eigenvector by applying the algorithm to the matrix $(\widetilde{\mathbf{A}} - \lambda_1 \mathbf{x}_1 \mathbf{x}_1^T)$.

A. Distributed implementation

For large-scale networks, the main computational burden is the matrix-vector product operation. Similar to the power iteration, it is possible to implement the MAPS algorithm in a distributed and parallel way such that the running time and/or complexity can be reduced. For such distributed implementations, the computational complexity per node depends on the size of its neighborhood.

V. Simulation Results

In this section, we present numerical results on the convergence behavior of the proposed algorithms. The errors between the estimate and the true value are defined as

$$\text{error}_{\text{eigenvalue}} = \log_{10}(|\widehat{\lambda}| - |\lambda|)^2, \tag{13}$$

$$\text{error}_{\text{eigenvector}} = \log_{10}(\|\mathbf{x}^{(t)} - \mathbf{x}\|_2^2). \tag{14}$$

We present the results in the form of error as a function of iteration number. The algorithms we consider here are the power method, Nesterov-Power, MAP, and MAPS.

The comparison is based on comparable computational complexity, meaning that for each iteration the product of a matrix and a vector is implemented only once for all algorithms. If the algorithms are implemented distributively over a graph by exchanging messages, then the communication loads of all algorithms are comparable. For the power iteration and the MAPS algorithm, since they do not need normalization at each iteration, their computational loads are less than the Nesterov-Power and the MAP algorithms.

We first randomly generate a graph with 100 nodes where the probability that any two nodes are connected is 5%. We then apply the various algorithms to the problem of finding the second largest eigenvalue and the corresponding eigenvector of the normalized adjacency matrix. The first eigenvalue was identified first, and a deflated matrix $(\widetilde{\mathbf{A}} - \lambda_1 \mathbf{x}_1 \mathbf{x}_1^T)$ is then considered for finding the second largest eigen pair. In Figure 1 and Figure 2, we depict the convergence behaviors of the eigenvalue and eigenvectors of the algorithms, respectively. It can be observed that all algorithms converge to the global optimal point and the proposed algorithms show faster convergence rates than the power method.

In the second example, we generate the matrix $\mathbf{A}$ by $\mathbf{A} = \mathbf{U}\Lambda\mathbf{U}^T$, where $\mathbf{U}$ is an unitary matrix, and Λ is a diagonal matrix whose diagonal entries are generated randomly and uniformly within $[0, 2]$. The dimension of the matrix is 100. The convergence behavior of the eigenvalue are presented in Figure 3. It can be seen that the differences among the Nesterov-Power, the MAP, and the MAPS algorithms are small. This also indicates that there is no need to normalize the vector at each iteration if the scalars d and ρ are chosen properly. Again, the proposed algorithms have a faster convergence rate than the well-known power method.

VI. Conclusions and Discussion

In this paper, we developed several numerical algorithms for improving the convergence speed of the power iteration for

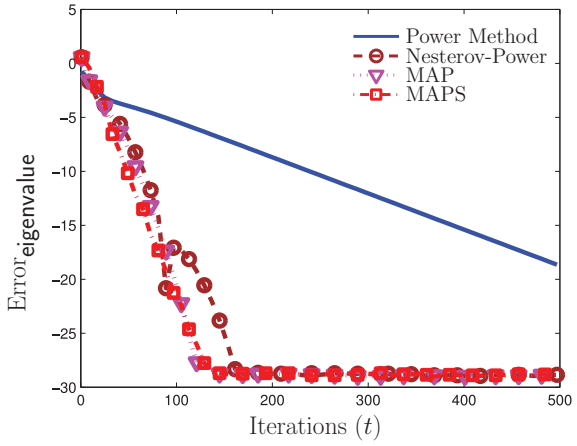

Fig. 1. The convergence behavior of the algorithms for spectral clustering, where $\beta^{(t)} = 0.7$ for Nesterov-Power, $\rho = 0.6$ for MAP, and $\rho = 1, \nu = 0.01, T' = 10$ for MAPS.

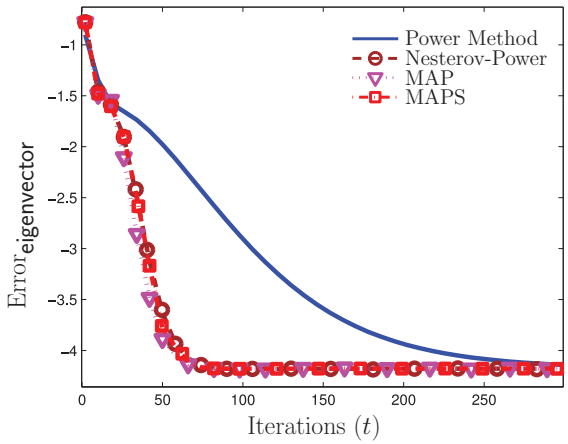

Fig. 2. The convergence behavior of the algorithms for spectral clustering, where $\beta^{(t)} = 0.7$ for Nesterov-Power, $\rho = 0.6$ for MAP, and $\rho = 1, \nu = 0.01, T' = 10$ for MAPS.

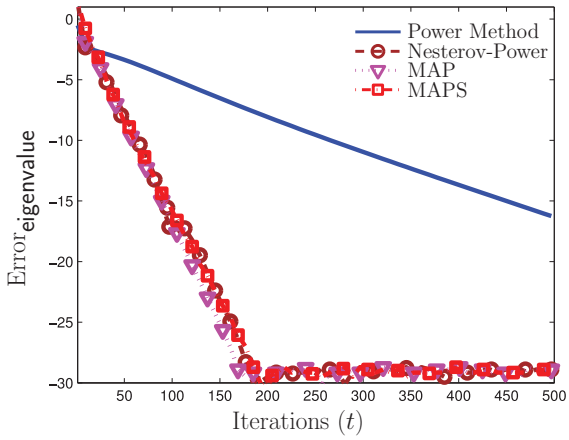

Fig. 3. The convergence behavior of the algorithms for eigen-decomposition problem, where $\beta^{(t)} = 0.7$ for Nesterov-Power, $\rho = 0.6$ for MAP, and $\rho = 1, \nu = 0.1, T' = 10$ for MAPS.

finding the leading eigen-pairs of a given matrix, including the MAP, MAPS, and Nesterov-Power methods. The methods we presented all introduce a memory term, although in slightly different fashions. The memory terms enable the algorithms to take advantage of the iterates that have already been performed. For the MAPS algorithm, global convergence guarantee and convergence rate results are also presented. We briefly discussed the possible application of the proposed algorithms in spectral clustering problems, possibly with distributed implementations. Simulation results showed that the proposed algorithms enjoy a faster convergence rate compared to the well-known power iteration.

We remark that it is possible to combine the proposed memory-based modifications with other techniques such as the shift power method. Also extension of the proposed method to the simultaneous calculation of multiple extreme eigen-pairs along the lines of [5] would be interesting.

Acknowledgment: The work in this paper was supported in part by NSF Grants No. 1523374 and No. 1308419. The authors would like to thank Georgios Giannakis for discussion on spectral clustering, and Qi Xiao and Mingyi Hong for discussion on the convergence analysis.

APPENDIX

In this appendix, we provide the proof of Theorem 1. We can write

$$\mathbf{x}^{(0)} = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + \ldots + c_N\mathbf{x}_N \tag{15}$$

where $c_1, \ldots, c_N$ are coefficients, $\mathbf{A}\mathbf{x}_i = \lambda_i\mathbf{x}_i$ and $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_N|$. After the t iterations, due to the orthogonality of $\mathbf{x}_i, i = 1, \ldots, N$, we have

$$\mathbf{x}^{(t)} = c_1 s^{(t)}(\lambda_1)\mathbf{x}_1 + \ldots + c_N s^{(t)}(\lambda_N)\mathbf{x}_N \tag{16}$$

$$= [\mathbf{x}_1, \ldots, \mathbf{x}_N] \begin{bmatrix} c_1 s^{(t)}(\lambda_1) \\ \vdots \\ c_N s^{(t)}(\lambda_N) \end{bmatrix} \tag{17}$$

$$= \mathbf{U} \begin{bmatrix} c_1 s^{(t)}(\lambda_1) \\ \vdots \\ c_N s^{(t)}(\lambda_N) \end{bmatrix}$$

where $s^{(t)}(\lambda_i)$ is a function of λ_i and $\mathbf{U} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]$.

We can write the iteration in the MAPS algorithm as

$$\mathbf{x}^{(t+1)} = d\mathbf{A}\mathbf{x}^{(t)} - \rho\mathbf{x}^{(t-1)} \tag{18}$$

$$= d\mathbf{U}\Lambda\mathbf{U}^T\mathbf{x}^{(t)} - \rho\mathbf{x}^{(t-1)} \tag{19}$$

where Λ is a diagonal matrix whose diagonal entries are $\lambda_1, \ldots, \lambda_N$. Defining $\mathbf{z}^{(t)} = \mathbf{U}^T\mathbf{x}^{(t)}$, we have

$$\mathbf{z}^{(t+1)} = d\Lambda\mathbf{z}^{(t)} - \rho\mathbf{z}^{(t-1)} \tag{20}$$

For each entry of $\mathbf{z}^{(t)}$, we have

$$\mathbf{z}_i^{(t+1)} = d\lambda_i\mathbf{z}_i^{(t)} - \rho\mathbf{z}_i^{(t-1)}. \tag{21}$$

As a result, the recurrence of $\mathbf{z}_i^{(t)}$ is equivalent to the recurrence of $s^{(t)}(\lambda_i)$. Specifically, substituting (16) into $\mathbf{z}^{(t)} = \mathbf{U}^T\mathbf{x}^{(t)}$, we have

$$\mathbf{z}^{(t)} = \mathbf{U}^T\mathbf{U}\begin{bmatrix} c_1 s^{(t)}(\lambda_1) \\ \vdots \\ c_N s^{(t)}(\lambda_N) \end{bmatrix} = \begin{bmatrix} c_1 s^{(t)}(\lambda_1) \\ \vdots \\ c_N s^{(t)}(\lambda_N) \end{bmatrix}, \quad (22)$$

which means that

$$s^{(t+1)}(\lambda_i) = d\lambda_i s^{(t)}(\lambda_i) - \rho^{(t-1)}(\lambda_i). \quad (23)$$

We next discuss the convergence of (16) when t is large enough. Define

$$a_i = d\lambda_i, \quad (24)$$

$$\gamma_i = \sqrt{a_i^2 - 4\rho} \quad (25)$$
$$= \sqrt{d^2\lambda_i^2 - 4\rho}.$$

There are two cases for the closed-form expression of $s^{(t)}(\lambda_i)$, as follows.

1) If $\gamma_i \neq 0, \forall i$, we have

$$s^{(t)}(\lambda_i) = \frac{1}{\gamma_i}\left[\left(-\rho + \frac{a_i+\gamma_i}{2}\right)\left(\frac{a_i+\gamma_i}{2}\right)^t - \left(-\rho + \frac{a_i-\gamma_i}{2}\right)\left(\frac{a_i-\gamma_i}{2}\right)^t\right]. \quad (26)$$

For the further simplification of the expression (26), define the coefficients

$$\xi_i^{(+)} = \frac{1}{\gamma_i}\left(-\rho + \frac{a_i+\gamma_i}{2}\right), \quad (27)$$

$$\xi_i^{(-)} = \frac{1}{\gamma_i}\left(-\rho + \frac{a_i-\gamma_i}{2}\right). \quad (28)$$

We can rewrite (16) as

$$\mathbf{x}^{(t)} = \quad (29)$$
$$\sum_{i=1}^{N}\left(c_i\xi_i^{(+)}\left(\frac{a_i+\gamma_i}{2}\right)^t - c_i\xi_i^{(-)}\left(\frac{a_i-\gamma_i}{2}\right)^t\right)\mathbf{x}_i$$
$$= c_1\xi_1^{(+)}\left(\frac{d\lambda_1 + \sqrt{d^2\lambda_1^2 - 4\rho}}{2}\right)^t$$
$$\cdot\left(\left(1 - \frac{\xi_1^{(-)}}{\xi_1^{(+)}}\left(\frac{d\lambda_1 - \sqrt{d^2\lambda_1^2 - 4\rho}}{d\lambda_1 + \sqrt{d^2\lambda_1^2 - 4\rho}}\right)^t\right)\mathbf{x}_1\right.$$
$$+ \sum_{i=2}^{N}\left(\frac{c_i\xi_i^{(-)}}{c_1\xi_1^{(+)}}\right)\left(\frac{d\lambda_i + \sqrt{d^2\lambda_i^2 - 4\rho}}{d\lambda_1 + \sqrt{d^2\lambda_1^2 - 4\rho}}\right)^t\mathbf{x}_i$$
$$\left.- \sum_{i=2}^{N}\left(\frac{c_i\xi_i^{(-)}}{c_1\xi_1^{(+)}}\right)\left(\frac{d\lambda_i - \sqrt{d^2\lambda_i^2 - 4\rho}}{d\lambda_1 + \sqrt{d^2\lambda_1^2 - 4\rho}}\right)^t\mathbf{x}_i\right).$$

For notational convenience, we define

$$\psi_i^{(+)} = d\lambda_i + \sqrt{d^2\lambda_i^2 - 4\rho}, \quad (30)$$

$$\psi_i^{(-)} = d\lambda_i - \sqrt{d^2\lambda_i^2 - 4\rho}. \quad (31)$$

Then, the requirements on $\psi_i^{(+)}$ and $\psi_i^{(-)}$ for convergence of the MAPS algorithm are

a) $|\psi_i^{(+)}| < |\psi_1^{(+)}|$, $i = 2, \ldots, N$;
b) $|\psi_i^{(-)}| < |\psi_1^{(+)}|$, $i = 1, 2, \ldots, N$;
c) $|\frac{\psi_1^{(+)}}{2}| > 1$.

A sufficient condition for the convergence is

$$d > \frac{\rho+1}{|\lambda_1|}. \quad (32)$$

2) If $\gamma_i = 0$ for some i, we have

$$s^{(t)}(\lambda_i) = \left(\frac{d\lambda_i}{2}\right)^t\left[\left(\frac{2}{d} - 1\right)t + 1\right]. \quad (33)$$

It can be seen that for $i \geq 2$, $s^{(t)}(\lambda_i)$ is dominated by $s^{(t)}(\lambda_1)$ for large t. So $s^{(t)}(\lambda_i)$ does not cause divergence of the algorithm if $\gamma_i = 0$. If $\gamma_1 = 0$, it can be shown that a sufficient condition of the MAPS iteration to converge to $\mathbf{x}_1$ is $d \neq 2$. If we have

$$d > \frac{\rho+1}{|\lambda_1|} \geq \frac{2\sqrt{\rho}}{|\lambda_1|}, \quad (34)$$

the case $\gamma_1 = 0$ can be avoided if we choose $d > \frac{\rho+1}{|\lambda_1|}$.

In summary, a sufficient condition for the algorithm to converge to the global optimal solution is $d > \frac{\rho+1}{|\lambda_1|}$.

REFERENCES

[1] J. H. Wilkinson, *The algebraic eigenvalue problem*, vol. 87, Clarendon Press, Oxford, 1965.

[2] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, 2007.

[3] P. Traganitis, K. Slavakis, and G. Giannakis, "Spectral clustering of large-scale communities via random sketching and validation," in *Proc. of Conf. Inf. Sci. Syst. (CISS)*, Mar. 2015.

[4] P. Traganitis, K. Slavakis, and G. Giannakis, "Sketch and validate for big data clustering," *IEEE J. Sel. Topics Signal Process*, vol. 9, no. 4, pp. 678–690, June 2015.

[5] J. E. Gubernatis and T. E. Booth, "Multiple extremal eigenpairs by the power method," *Journal of Computational Physics*, vol. 227, no. 19, pp. 8508–8522, 2008.

[6] W. E. Arnoldi, "The principle of minimized iterations in the solution of the matrix eigenvalue problem," *Quarterly of Applied Mathematics*, vol. 9, pp. 17–29, 1951.

[7] C. Lanczos, "An iteration method for the solution of the eigenvalue problem of linear differential and integral operators," *Journal of Research of the National Bureau of Standards*, vol. 45, no. 4, pp. 255–282, 1950.

[8] Y. Deshpande, A. Montanari, and E. Richard, "Cone-constrained principal component analysis," in *Proc. of Advances in Neural Information Processing Systems (NIPS)*, pp. 2717–2725, 2014.

[9] A. Montanari and E. Richard, "Non-negative principal component analysis:message passing algorithms and sharp asymptotics," *IEEE Trans. Info. Theory*, 2015.

[10] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," in *Soviet Mathematics Doklady*, vol. 9, pp. 372–376, 1983.