ONLINE DOMINANT GENERALIZED EIGENVECTORS EXTRACTION VIA A RANDOMIZED METHOD

Haoyuan Cai [*], *Maboud F. Kaloorazi* [*], *Jie Chen* [*], *Wei Chen* [†] *and Cédric Richard* [‡]

[*] Center of Intelligent Acoustics and Immersive Communications (CIAIC)
School of Marine Science and Technology, Northwestern Polytechnical University, China
[†] State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, China
[†] Univiersité Cote d'Azur, France
Emails: haoyuan.cai@mail.nwpu.edu.cn, kaloorazi@nwpu.edu.cn, dr.jie.chen@ieee.org, weich@bjtu.edu.cn, cedric.richard@unice.fr

ABSTRACT

The generalized Hermitian eigendecomposition problem is ubiquitous in signal and machine learning applications. Considering the need of processing streaming data in practice and restrictions of existing methods, this paper is concerned with fast and efficient generalized eigenvectors tracking. We first present a computationally efficient algorithm based on randomization termed alternate-projections randomized eigenvalue decomposition (APR-EVD) to solve a standard eigenvalue problem. By exploiting rank-1 strategy, two online algorithms based on APR-EVD are developed for the dominant generalized eigenvectors extraction. Numerical examples show the practical applicability and efficacy of the proposed online algorithms.

***Index Terms*—** Randomized algorithms, dominant generalized eigenvectors, online algorithms, fast subspace tracking.

1. INTRODUCTION

The generalized Hermitian eigenvalue problem (GHEP) [1] is of great interest in signal processing, machine learning and data analysis applications. The GHEP algorithms provide powerful tools to treat problems in blind source separation [2,3], feature extraction [4, 5], noise filtering [6], fault detection [7], antenna array processing [8], classification [9], and speech enhancement [10]. Traditional methods for solving the GHEP include power and inverse iteration based methods, Lanczos method and Jacobi-Davidson method [1, 11]. These batch methods, however, are inefficient and, in some cases, infeasible to apply due to their computational workload. The online methods presented in [8,9,12] are gradient-based, and extract the first dominant (or principal) generalized eigenvector. However, they are unsuitable for applications where multiple dominant generalized eigenvectors are desired [10]. In addition, these methods suffer from the so-called speed-stability problem [13], i.e., it is hard to select an appropriate learning rate to guarantee tracking speed and numerical stability. To address the issue, coupled learning methods were proposed in [14, 15]. These methods, which are considered as sequential methods, in addition to be difficult to parallelize in order to harness modern computational platforms, may even cause error propagation during the procedure of orthogonal projection. Yang et al. [16] proposed recursive least-square (RLS)-based online algorithms based on the projection approximation subspace tracking (PAST) technique [17]. The work in [18] presented a computationally efficient algorithm, but it suffers from slow convergence speed because of narrow search space [19]. Tanaka [19] developed an online algorithm based on the power iteration scheme. To track the r-dominant generalized eigenvectors, however, this method needs $\mathcal{O}(rN^2)$ operations in each iteration (with N being the dimension of an input matrix), which is still computationally expensive.

The generalized eigenvalues and eigenvectors are extracted from a *matrix pencil* $(\mathbf{A}, \mathbf{B})$. In online applications [2, 4, 6–10], however, this pair is unknown, and the rank-1 update strategy [14–16, 18, 19] uses the observed streaming stochastic signals to estimate it. Also, in many cases, the signal subspace spanned by the dominant generalized eigenvectors, lies in a low-dimensional space [10]. This implies that low-rank approximation techniques can be applied to treat GHEPs. Recent low-rank approximation methods based on randomized sampling [20–22] are computationally efficient and, in addition, can harness advanced computer architectures.

Our Contributions. Through compounding a randomized low-rank matrix factorization method and the rank-1 update strategy, we propose two online algorithms for r-dominant generalized eigenvectors extraction, where $r \geq 1$: we first present the APR-EVD (alternate-projections randomized eigenvalue decomposition) algorithm that efficiently solves a standard eigenvalue problem. Then, by harnessing the rank-1 update scheme, we devise two line algorithms to extract the generalized eigenvectors with streaming data. Our proposed algorithms are computationally efficient, as the necessary steps in each iteration need $\mathcal{O}(N^2)$ operations. Further, they can be parallelized on modern computers.

Notation. Normal fonts x and X denote scalar. Boldface small letters $\mathbf{x}$ and capital letters $\mathbf{X}$ denote column vectors and matrices, respectively. $\mathbb{C}$ denotes the complex domain. The superscript $(\cdot)^*$ denotes the conjugate of a complex number, $(\cdot)^{\mathrm{H}}$ denotes the Hermitian transpose operator, and $(\cdot)^\dagger$ denotes the pseudo-inverse of a matrix. $\mathbf{I}_N$ denotes an identity matrix of order N. $\mathtt{orth}_r(\cdot)$ constructs an orthonormal basis with r columns for the range of a matrix.

2. PROBLEM FORMULATION

Given a matrix pencil $(\mathbf{R}_y, \mathbf{R}_x)$, where $\mathbf{R}_y$, $\mathbf{R}_x \in \mathbb{C}^{N\times N}$ are Hermitian and positive definite, the GHEP [1, 11] is defined as:

$$\mathbf{R}_y\mathbf{w}_i = \lambda_i\mathbf{R}_x\mathbf{w}_i, \quad i = 1, \cdots, N \tag{1}$$

where $\mathbf{w}_1, \cdots, \mathbf{w}_N$ are nonzero vectors corresponding to N generalized eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_N > 0$. Provided that $\mathbf{R}_x$ is invertible, to obtain a generalized eigen-pair $(\mathbf{w}_i, \lambda_i)$, in general, (1) is reduced to a Hermitian or non-Hermitian eigenvalue problem:

This work was supported in part by NSFC grants 61671382 and 61811530283, and 111 project (B18041). Corresponding author: J. Chen.

`Case HEP`: Provided that $\mathbf{R}_x^{1/2}(\mathbf{R}_x^{1/2})^{\mathrm{H}} = \mathbf{R}_x$, the set of eigenvectors $\mathbf{w}$ is obtained by $(\mathbf{R}_x^{-1/2})^{\mathrm{H}}\mathbf{v}$, where $\mathbf{v}$ is the set of eigenvectors of $\mathbf{R}_x^{-1/2}\mathbf{R}_y(\mathbf{R}_x^{-1/2})^{\mathrm{H}}$ determined so that $\mathbf{v}^T\mathbf{R}_x\mathbf{v} = \mathbf{I}$.
`Case Non-HEP`: The generalized eigenvectors $\mathbf{w}_i$ and generalized eigenvalues λ_i are obtained by solving $\mathbf{R}_x^{-1}\mathbf{R}_y\mathbf{w}_i = \lambda_i\mathbf{w}_i$.

In many signal and information processing applications, $\mathbf{R}_x$ and $\mathbf{R}_y$ are associated to data covariance matrices. Let the covariance matrices of $\mathbf{x}(k)$ and $\mathbf{y}(k)$ be given by $\mathbf{R}_x = \mathbb{E}\{\mathbf{x}(k)\mathbf{x}^{\mathrm{H}}(k)\}$ and $\mathbf{R}_y = \mathbb{E}\{\mathbf{y}(k)\mathbf{y}^{\mathrm{H}}(k)\}$. When processing streaming data, at each instant k these matrices are typically estimated by time averaging with the most recent data [14–16, 18, 19, 23] as follows:

$$\mathbf{R}_x(k) = \alpha\mathbf{R}_x(k-1) + \mathbf{x}(k)\mathbf{x}^{\mathrm{H}}(k), \tag{2}$$

$$\mathbf{R}_y(k) = \beta\mathbf{R}_y(k-1) + \mathbf{y}(k)\mathbf{y}^{\mathrm{H}}(k), \tag{3}$$

where parameters $\alpha \in (0,1)$ and $\beta \in (0,1)$ are smoothing constants.

Under the above setting, in this paper we devise two online algorithms by first transforming (1) into a standard eigenvalue problem and then apply a randomized EVD algorithm, which enables processing streaming data for tracking generalized eigenvactors.

3. PROPOSED ALGORITHMS

In this section, we first propose a randomized eigenvalue decomposition algorithm. We then adapt this algorithm to the streaming data setting with `Case HEP` and `Case Non-HEP` respectively.

3.1. The APR-EVD Algorithm

Given a rank-r matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$, the proposed APR-EVD algorithm is computed as follows: we generate a random Gaussian matrix $\mathbf{\Psi} \in \mathbb{C}^{N\times d}$, where $r \le d < N$. Then, we construct the matrix:

$$\mathbf{G} = \mathbf{A}^{\mathrm{H}}\mathbf{\Psi}. \tag{4}$$

Matrix $\mathbf{G} \in \mathbb{C}^{N\times d}$ is a projection onto the row space of $\mathbf{A}$ by $\mathbf{\Psi}$. Next, we form the $N \times d$ matrix:

$$\mathbf{H} = \mathbf{AG}. \tag{5}$$

Matrix $\mathbf{H}$ is a projection onto the column space of $\mathbf{A}$ by $\mathbf{G}$. After, we orthonormalize the columns of $\mathbf{H}$ to obtain an $N \times r$ basis $\mathbf{Q}$:

$$\mathbf{Q} = \mathtt{orth}_r(\mathbf{H}). \tag{6}$$

Note that the rank of $\mathbf{H}$ is at most r [24], and $\mathbf{Q}$ approximates the range of $\mathbf{A}$. Through exploiting $\mathbf{Q}$, we use the Rayleigh-Ritz method [25, 26] to compute the eigenvalues λ_i and corresponding eigenvectors $\mathbf{v}_i$ of the following matrix which is of order r:

$$\mathbf{T} = \mathbf{Q}^{\mathrm{H}}\mathbf{AQ}. \tag{7}$$

Defining $\mathbf{u}_i \triangleq \mathbf{Q}\mathbf{v}_i$, consequently $\{(\mathbf{u}_i, \lambda_i)\}_{i=1}^r$ constitute the approximate eigen-pairs of $\mathbf{A}$.

APR-EVD makes two passes over $\mathbf{A}$ given the matrix is stored in the row-major format. By approximating $\mathbf{T}$ in (7), we devise a single-pass algorithm, which can be directly employed for streaming data processing. In doing so, we pre-multiply (7) by $\mathbf{\Psi}^{\mathrm{H}}\mathbf{Q}$, obtaining $\mathbf{\Psi}^{\mathrm{H}}\mathbf{QT} = \mathbf{\Psi}^{\mathrm{H}}\mathbf{QQ}^{\mathrm{H}}\mathbf{AQ}$. Having known that $\mathbf{A} \approx \mathbf{QQ}^{\mathrm{H}}\mathbf{A}$ and by the definition of $\mathbf{G}$ (4), an estimate $\widetilde{\mathbf{T}}$ is given by:

$$\widetilde{\mathbf{T}} = (\mathbf{\Psi}^{\mathrm{H}}\mathbf{Q})^{\dagger}\mathbf{G}^{\mathrm{H}}\mathbf{Q}. \tag{8}$$

Computational Cost of APR-EVD. To factor $\mathbf{A}$, APR-EVD incurs these costs: generating a random matrix costs $\mathcal{O}(Nd)$. Forming $\mathbf{G}$ and $\mathbf{H}$ each costs $\mathcal{O}(N^2d)$. Considering an estimation to (7), forming $\widetilde{\mathbf{T}}$ and computing an eigenpair cost $\mathcal{O}(Nr^2)+\mathcal{O}(r^3)$. Thus, the operation count (dominated by multiplications of A) satisfies

$$\mathcal{C}_{\text{APR-EVD}} = \mathcal{O}(N^2d). \tag{9}$$

Here d (the sampling size parameter) is very close to r.

3.2. Online Algorithm for Extracting r-dominant Generalized Eigenvectors with `Case HEP` (Algorithm 1).

Directly recalculating $\mathbf{R}_x^{-1/2}(k)\mathbf{R}_y(k)(\mathbf{R}_x^{-1/2}(k))^{\mathrm{H}}$ has the computation complexity of $\mathcal{O}(N^3)$. We therefore consider the estimated covariance matrices by rank-1 matrices at each instant, i.e., equations (2) and (3), together with the proposed APR-EVD algorithm to recursively compute the generalized eigenvectors. For ease of notation, let $\mathbf{K}(k) = \mathbf{R}_x^{-1/2}(k)$ and $\mathbf{R}(k) = \mathbf{K}(k)\mathbf{R}_y(k)\mathbf{K}^{\mathrm{H}}(k)$.

In order to use APR-EVD with the arrival of $\mathbf{x}(k)$ and $\mathbf{y}(k)$, we recursively update $\mathbf{R}(k)$, then $\mathbf{G}(k) = \mathbf{R}^{\mathrm{H}}(k)\mathbf{\Psi}(k)$ and $\mathbf{H}(k) = \mathbf{R}(k)\mathbf{G}(k)$ so that the orthonormal basis at instant k, $\mathbf{Q}(k)$, can be extracted. Exploiting the results in [19], we update $\mathbf{R}(k)$ and $\mathbf{K}(k)$:

$$\begin{aligned}\mathbf{R}(k) = \frac{1}{\alpha}\big[&\beta\mathbf{R}(k-1) + \tilde{\mathbf{y}}(k)\tilde{\mathbf{y}}^{\mathrm{H}}(k) + \tilde{\mathbf{x}}(k)\mathbf{c}^{\mathrm{H}}(k)\\ &+ \delta_1(k)\mathbf{h}(k)\tilde{\mathbf{x}}^{\mathrm{H}}(k)\big],\\ \mathbf{K}(k) = \frac{1}{\sqrt{\alpha}}&\mathbf{K}(k-1) + \delta_1(k)\tilde{\mathbf{x}}(k)\bar{\mathbf{x}}^{\mathrm{H}}(k).\end{aligned} \tag{10}$$

where

$$\tilde{\mathbf{y}}(k) = \mathbf{K}(k-1)\mathbf{y}(k), \tag{11}$$

$$\tilde{\mathbf{x}}(k) = \frac{1}{\sqrt{\alpha}}\mathbf{K}(k-1)\mathbf{x}(k), \tag{12}$$

$$\mathbf{c}(k) = \delta_2^*(k)\tilde{\mathbf{x}}(k) + \delta_1(k)\mathbf{h}(k), \tag{13}$$

$$\mathbf{h}(k) = \beta\mathbf{r}_x(k) + a_1(k)\tilde{\mathbf{y}}(k), \tag{14}$$

$$\bar{\mathbf{x}}(k) = \frac{1}{\sqrt{\alpha}}\mathbf{K}^{\mathrm{H}}(k-1)\tilde{\mathbf{x}}(k). \tag{15}$$

In the above relations, $a_1(k)$, $\delta_1(k)$, $\delta_2(k)$ and $\mathbf{r}_x(k)$ are defined by:

$$a_1(k) = \tilde{\mathbf{y}}^{\mathrm{H}}(k)\tilde{\mathbf{x}}(k), \tag{16}$$

$$\delta_1(k) = \frac{1}{\|\tilde{\mathbf{x}}(k)\|^2}\Big(\frac{1}{\sqrt{1+\|\tilde{\mathbf{x}}(k)\|^2}} - 1\Big), \tag{17}$$

$$\delta_2(k) = |\delta_1(k)|^2\,(\beta\tilde{\mathbf{x}}^{\mathrm{H}}(k)\mathbf{r}_x(k) + |a_1(k)|^2), \tag{18}$$

$$\mathbf{r}_x(k) = \mathbf{R}(k-1)\tilde{\mathbf{x}}(k). \tag{19}$$

After updating $\mathbf{R}(k)$ and $\mathbf{K}(k)$, $\mathbf{G}(k)$ is obtained by the recursion:

$$\begin{aligned}\mathbf{G}(k) =&\mathbf{R}^{\mathrm{H}}(k)\mathbf{\Psi}\\ =&\frac{1}{\alpha}\big[\beta\mathbf{G}(k-1) + \tilde{\mathbf{y}}(k)\mathbf{y}_o^{\mathrm{H}}(k) + \mathbf{c}(k)\mathbf{x}_o^{\mathrm{H}}(k)\\ &+ \delta_1(k)\tilde{\mathbf{x}}(k)\mathbf{h}_o^{\mathrm{H}}(k)\big],\end{aligned} \tag{20}$$

where $\mathbf{y}_o(k) \triangleq \mathbf{\Psi}^{\mathrm{H}}\tilde{\mathbf{y}}(k)$, $\mathbf{x}_o(k) \triangleq \mathbf{\Psi}^{\mathrm{H}}\tilde{\mathbf{x}}(k)$, and $\mathbf{h}_o(k) \triangleq \mathbf{\Psi}^{\mathrm{H}}\mathbf{h}(k)$. After that, $\mathbf{H}(k)$ is obtained through the recursion:

$$\begin{aligned}\mathbf{H}(k) &= \mathbf{R}(k)\mathbf{G}(k)\\ &= \frac{1}{\alpha^2}\big[\beta^2\mathbf{H}(k-1) + \mathbf{S}_1(k) + \mathbf{S}_2(k) + \mathbf{S}_3(k) + \mathbf{S}_4(k)\big].\end{aligned} \tag{21}$$

The terms $\{\mathbf{S}_i(k)\}_{i=1}^4$ are given by:

$$\mathbf{S}_1(k) = \beta \mathbf{r}_y(k)\mathbf{y}_o^{\mathrm{H}}(k) + \beta \mathbf{r}_c(k)\mathbf{x}_o^{\mathrm{H}}(k) + \beta\delta_1(k)\mathbf{r}_x(k)\mathbf{h}_o^{\mathrm{H}}(k), \tag{22}$$

$$\begin{aligned}\mathbf{S}_2(k) = \tilde{\mathbf{y}}(k)\big[&\beta \mathbf{y}_h^{\mathrm{H}}(k) + a_2(k)\mathbf{y}_o^{\mathrm{H}}(k) \\ &+ a_3^*(k)\mathbf{x}_o^{\mathrm{H}}(k) + \delta_1(k)a_1(k)\mathbf{h}_o^{\mathrm{H}}(k)\big],\end{aligned} \tag{23}$$

$$\begin{aligned}\mathbf{S}_3(k) = \tilde{\mathbf{x}}(k)\big[&\beta \mathbf{c}_h^{\mathrm{H}}(k) + a_3(k)\mathbf{y}_o^{\mathrm{H}}(k) \\ &+ a_4(k)\mathbf{x}_o^{\mathrm{H}}(k) + \delta_1(k)a_5(k)\mathbf{h}_o^{\mathrm{H}}(k)\big],\end{aligned} \tag{24}$$

$$\begin{aligned}\mathbf{S}_4(k) = \mathbf{h}(k)\big[&\delta_1(k)\beta\mathbf{x}_h^{\mathrm{H}}(k) + \delta_1(k)a_1^*(k)\mathbf{y}_o^{\mathrm{H}}(k) \\ &+ \delta_1(k)a_5^*(k)\mathbf{x}_o^{\mathrm{H}}(k) + |\delta_1(k)|^2\, a_6(k)\mathbf{h}_o^{\mathrm{H}}(k)\big].\end{aligned} \tag{25}$$

where

$$\mathbf{r}_y(k) = \mathbf{R}(k-1)\tilde{\mathbf{y}}(k), \tag{26}$$
$$\mathbf{r}_c(k) = \mathbf{R}(k-1)\mathbf{c}(k), \tag{27}$$
$$\mathbf{y}_h(k) = \mathbf{G}^{\mathrm{H}}(k-1)\tilde{\mathbf{y}}(k), \tag{28}$$
$$a_2(k) = \tilde{\mathbf{y}}^{\mathrm{H}}(k)\tilde{\mathbf{y}}(k), \tag{29}$$
$$a_3(k) = \mathbf{c}^{\mathrm{H}}(k)\tilde{\mathbf{y}}(k), \tag{30}$$
$$\mathbf{c}_h(k) = \mathbf{G}^{\mathrm{H}}(k-1)\mathbf{c}(k), \tag{31}$$
$$a_4(k) = \mathbf{c}^{\mathrm{H}}(k)\mathbf{c}(k), \tag{32}$$
$$a_5(k) = \mathbf{c}^{\mathrm{H}}(k)\tilde{\mathbf{x}}(k), \tag{33}$$
$$\mathbf{x}_h(k) = \mathbf{G}^{\mathrm{H}}(k-1)\tilde{\mathbf{x}}(k), \tag{34}$$
$$a_6(k) = \mathbf{x}^{\mathrm{H}}(k)\tilde{\mathbf{x}}(k). \tag{35}$$

Next, we orthonormalize the columns of $\mathbf{H}(k)$ (21), obtaining $\mathbf{Q}(k)$:

$$\mathbf{Q}(k) = \mathtt{orth}_r(\mathbf{H}(k)), \tag{36}$$

Then, $\widetilde{\mathbf{T}}(k)$ is computed through the formula (8):

$$\widetilde{\mathbf{T}}(k) = (\mathbf{\Psi}^H\mathbf{Q}(k))^\dagger(\mathbf{G}^H(k)\mathbf{Q}(k)). \tag{37}$$

By performing the Rayleigh-Ritz method on $\widetilde{\mathbf{T}}(k)$, we obtain the eigenpair $(\widetilde{\mathbf{\Lambda}}(k), \widetilde{\mathbf{V}}(k))$. Accordingly, an approximation to the r leading generalized eigenvectors of $(\mathbf{R}_y, \mathbf{R}_x)$ is obtained by:

$$\mathbf{W}_r(k) = \mathbf{K}(k)\mathbf{Q}(k)\widetilde{\mathbf{V}}(k). \tag{38}$$

Computational Cost of Algorithm 1. The main steps involve computations of (10)-(35) in each iteration. The calculations of parameters $\{a_i(k)\}_{i=1}^6$, $\{\delta_i(k)\}_{i=1}^2$ require $\mathcal{O}(N)$ operations. Computing $\mathbf{h}(k)$, $\mathbf{c}(k) \in \mathbf{C}^N$ costs $\mathcal{O}(N)$. Computing $\mathbf{y}_h(k)$, $\mathbf{x}_h(k)$, $\mathbf{c}_h(k)$, $\mathbf{y}_o(k)$, $\mathbf{x}_o(k)$, $\mathbf{c}_o(k) \in \mathbb{C}^d$ for $\{\mathbf{S}_i(k)\}_{i=1}^4 \in \mathbb{C}^{N\times d}$, and updating $\mathbf{G}(k)$, $\mathbf{H}(k) \in \mathbf{C}^{N\times d}$ cost $\mathcal{O}(Nd)$. Computing the column vectors $\tilde{\mathbf{x}}(k)$, $\tilde{\mathbf{y}}(k)$, $\bar{\mathbf{x}}(k)$, $\mathbf{r}_y(k)$, $\mathbf{r}_x(k)$, $\mathbf{r}_c(k) \in \mathbb{C}^N$ requires $6N^2$ multiplications. Updating $\mathbf{K}(k)$, $\mathbf{R}(k) \in \mathbb{C}^{N\times N}$ requires $4N^2$ multiplications. Thus, the dominant cost of Algorithm 1 is $10N^2 + \mathcal{O}(Nd)$.

3.3. Online Algorithm for Extracting r-dominant Generalized Eigenvectors with `Case Non-HEP` (Algorithm 2).

Let $\mathbf{P}(k) = \mathbf{Q}_x(k)\mathbf{R}_y(k)$, where $\mathbf{Q}_x(k) = \mathbf{R}_x^{-1}(k)$. We first recursively updates $\mathbf{P}(k)$. Applying the SM-formula (Sherman-Morrison-formula) [11] immediately leads to a recursion for $\mathbf{Q}_x(k)$:

$$\begin{aligned}\mathbf{Q}_x(k) &= \big[\alpha\mathbf{R}_x(k-1) + \mathbf{x}(k)\mathbf{x}^{\mathrm{H}}(k)\big]^{-1} \\ &= \frac{1}{\alpha}\Big[\mathbf{Q}_x(k-1) - \frac{\mathbf{q}_x(k)\mathbf{q}_x^{\mathrm{H}}(k)}{\alpha + \mathbf{x}^{\mathrm{H}}(k)\mathbf{q}_x(k)}\Big],\end{aligned} \tag{39}$$

where $\mathbf{q}_x(k) \triangleq \mathbf{Q}_x(k-1)\mathbf{x}(k)$. Using the above recursion and (2), $\mathbf{P}(k)$ is consequently obtained by:

$$\mathbf{P}(k) = \frac{1}{\alpha}\big[\beta\mathbf{P}(k-1) + \mathbf{q}_y(k)\mathbf{y}^{\mathrm{H}}(k) - \mathbf{q}_x(k)\mathbf{z}^{\mathrm{H}}(k)\big], \tag{40}$$

where

$$\mathbf{q}_y(k) = \mathbf{Q}_x(k-1)\mathbf{y}(k), \tag{41}$$
$$\mathbf{z}(k) = \big[\frac{\beta\mathbf{x}^{\mathrm{H}}(k)\mathbf{P}(k-1)}{\alpha + \mathbf{x}^{\mathrm{H}}(k)\mathbf{q}_x(k)} + \frac{\mathbf{q}_x^{\mathrm{H}}(k)\mathbf{y}(k)\mathbf{y}^{\mathrm{H}}(k)}{\alpha + \mathbf{x}^{\mathrm{H}}(k)\mathbf{q}_x(k)}\big]^{\mathrm{H}}. \tag{42}$$

Combining these results leads to the update of $\mathbf{G}(k) = \mathbf{P}^{\mathrm{H}}(k)\mathbf{\Psi}$ as:

$$\mathbf{G}(k) = \frac{1}{\alpha}\Big[\beta\mathbf{G}(k-1) + \mathbf{y}(k)\mathbf{m}_y^{\mathrm{H}}(k) - \mathbf{z}(k)\mathbf{m}_x^{\mathrm{H}}(k)\Big], \tag{43}$$

where $\mathbf{m}_x(k) \triangleq \mathbf{\Psi}^{\mathrm{H}}\mathbf{q}_x(k)$ and $\mathbf{m}_y(k) \triangleq \mathbf{\Psi}^{\mathrm{H}}\mathbf{q}_y(k)$. After, $\mathbf{H}(k)$ is obtained through the following recursion:

$$\begin{aligned}\mathbf{H}(k) =& \mathbf{P}(k)\mathbf{G}(k) \\ =& \frac{1}{\alpha^2}\big[\beta^2\mathbf{H}(k-1) + \mathbf{J}_1(k) + \mathbf{J}_2(k) + \mathbf{J}_3(k)\big],\end{aligned} \tag{44}$$

The terms $\{\mathbf{J}_i(k)\}_{i=1}^3$ in (44) are given by:

$$\mathbf{J}_1(k) = \beta\mathbf{d}_y(k)\mathbf{m}_y^{\mathrm{H}}(k) - \beta\mathbf{d}_z(k)\mathbf{m}_x^{\mathrm{H}}(k), \tag{45}$$
$$\mathbf{J}_2(k) = \mathbf{q}_y(k)\big(\beta\mathbf{n}_y^{\mathrm{H}}(k) + b_1(k)\mathbf{m}_y^{\mathrm{H}}(k) - b_2(k)\mathbf{m}_x^{\mathrm{H}}(k)\big), \tag{46}$$
$$\mathbf{J}_3(k) = \mathbf{q}_x(k)\big(b_3(k)\mathbf{m}_x^{\mathrm{H}}(k) - \beta\mathbf{n}_z^{\mathrm{H}}(k) - b_2^*(k)\mathbf{m}_y^{\mathrm{H}}(k)\big), \tag{47}$$

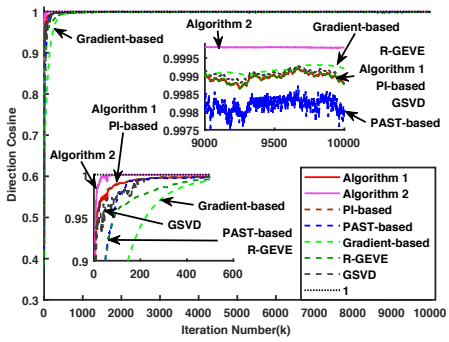
where

$$\mathbf{d}_y(k) = \mathbf{P}(k-1)\mathbf{y}(k), \tag{48}$$
$$\mathbf{d}_z(k) = \mathbf{P}(k-1)\mathbf{z}(k), \tag{49}$$
$$\mathbf{n}_y(k) = \mathbf{G}^{\mathrm{H}}(k-1)\mathbf{y}(k), \tag{50}$$
$$b_1(k) = \mathbf{y}^{\mathrm{H}}(k)\mathbf{y}(k), \tag{51}$$
$$b_2(k) = \mathbf{y}^{\mathrm{H}}(k)\mathbf{z}(k), \tag{52}$$
$$\mathbf{n}_z(k) = \mathbf{G}^{\mathrm{H}}(k-1)\mathbf{z}(k), \tag{53}$$
$$b_3(k) = \mathbf{z}^{\mathrm{H}}(k)\mathbf{z}(k). \tag{54}$$

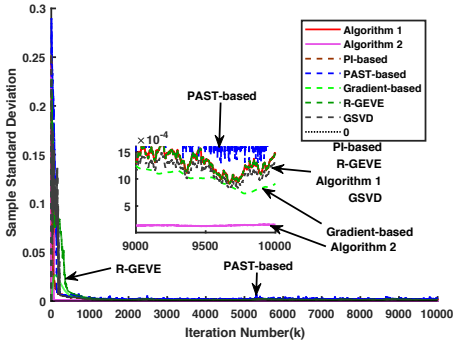
Following the procedure described in Algorithm 1, we form $\widetilde{\mathbf{T}}(k)$ and compute the eigenpair $(\widetilde{\mathbf{\Lambda}}(k), \widetilde{\mathbf{V}}(k))$. The r leading generalized eigenvectors of the matrix pencil $(\mathbf{R}_y, \mathbf{R}_x)$ are then estimated via:

$$\mathbf{W}_r(k) = \mathbf{Q}(k)\widetilde{\mathbf{V}}(k). \tag{55}$$

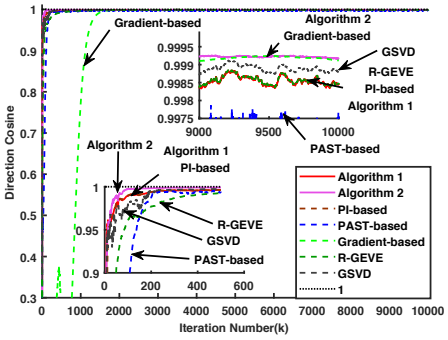
Computational Cost of Algorithm 2. Computing the parameters $\{b_i(k)\}_{i=1}^3$ requires $\mathcal{O}(N)$ operations. Computations of $\mathbf{m}_x(k)$, $\mathbf{m}_y(k)$, $\mathbf{n}_z(k)$, $\mathbf{n}_y(k) \in \mathbb{C}^d$ cost $\mathcal{O}(Nd)$. Computing $\{\mathbf{J}_i(k)\}_{i=1}^3 \in \mathbb{C}^{N\times d}$, and updating $\mathbf{G}(k)$, $\mathbf{H}(k) \in \mathbb{C}^{N\times d}$ cost $\mathcal{O}(Nd)$. Updating $\mathbf{q}_x(k)$, $\mathbf{q}_y(k)$, $\mathbf{d}_z(k)$, $\mathbf{d}_y(k)$, $\mathbf{z}(k) \in \mathbb{C}^N$ needs $5N^2$ multiplications, and calculations of $\mathbf{P}(k)$, $\mathbf{Q}_x(k)$ require $3N^2$ multiplications. The dominant cost of Algorithm 2 is thus $8N^2 + \mathcal{O}(Nd)$. The extraction of generalized eigenvectors needs $\mathcal{O}(Nr^2)$ +$\mathcal{O}(r^3)$ operations. Therefore, the flop count of Algorithm 2 satisfies $8N^2 + \mathcal{O}(Nr^2)$.



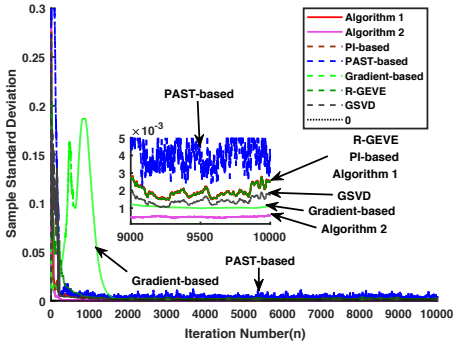
(a) Direction cosine of the 1st generalized eigenvector.



(b) Sample standard deviation of the 1st generalized eigenvector.



(c) Direction cosine of the 2nd generalized eigenvector.



(d) Sample standard deviation of the 2nd generalized eigenvector.

Fig. 1: DC and SSD results of tracking the first and second eigenvectors.

4. EXPERIMENTAL RESULTS

We validate proposed Algorithms 1 and 2 through a subspace tracking problem. The performance of our algorithms are compared with the following algorithms: (1) Power-iteration based (PI-based) method [19], (2) PAST-based method, here we use the sequential version [16, Algorithm 2], (3) reduced-rank generalized eigenvector extraction (R-GEVE) [18], (4) gradient-based method with negative step-size [15, Algorithm 3], and (5) batch mode GSVD [11].

Our signals are generated by two sinusoids in additive noise defined in the time domain [14–16, 18, 19, 23] as follows:

$$x(k)=\sqrt{2}\sin\left(0.46\pi k+\theta_2\right)+\sqrt{2}\sin\left(0.74\pi k+\theta_3\right)+n_1(k), \quad (56)$$

$$y(k)=\sqrt{2}\sin\left(0.62\pi k+\theta_1\right)+n_2(k), \quad (57)$$

where $\{\theta_i\}_{i=1}^3$ follow the uniform distribution $\mathcal{U}(0,2\pi)$, and $n_1(k)$ and $n_2(k)$ are zero-mean white Gaussian noises with variance $\sigma_1^2 = \sigma_2^2 = 0.1$. The vectors $\{\mathbf{y}(k)\}$ and $\{\mathbf{x}(k)\}$ are arranged in blocks of size $N = 8$, that is, $\mathbf{y}(k) = [y(k), \cdots, y(k-N+1)]^\top$, $\mathbf{x}(k) = [x(k), \cdots, x(k-N+1)]^\top$, and $k \geq N$. The generalized eigenvalues of matrix pencil $(\mathbf{R}_x, \mathbf{R}_y)$ are given by $\lambda_1 = 16.0680$, $\lambda_2 = 6.8302$, $\lambda_3 = 1.0$, $\lambda_4 = 1.0$, $\lambda_5 = 0.1592$, $\lambda_6 = 0.0708$, $\lambda_7 = 0.0254$, and $\lambda_8 = 0.0198$. We track the first four dominant generalized eigenvectors. For the random matrix $\mathbf{\Psi}$ of Algorithms 1 and 2, we set $d = 5$. The parameters of considered algorithms are set as follows: For the PAST-based algorithm, we set $\mu = 0.998$ as suggested in [16]; For R-GEVE, we set $\beta_1 = \beta_2 = 0.998$ as proposed in [18]. For other algorithms, we set $\alpha = \beta = 0.998$. In the Gradient-based algorithm, the step-size is set to $\eta = -0.0005 \in (2/(\lambda_N - \lambda_1), 0)$. All algorithms are initialized with $\mathbf{R}_x = \mathbf{R}_y = \mathbf{I}_N$, $\mathbf{w}_i(0) = \mathbf{e}_i$ for $i = 1, \cdots, 4$, where $\mathbf{e}_i$ is the ith column of $\mathbf{I}_N$.

To compare the convergence speed and estimation accuracy, we use the direction cosine, which measures the similarity between the ith estimated and exact generalized eigenvectors of $(\mathbf{R}_y, \mathbf{R}_x)$:

$$\mathrm{DC}_i(k) = \frac{\left|\widetilde{\mathbf{w}}_i^{\mathrm{H}}(k)\mathbf{w}_i\right|}{\|\widetilde{\mathbf{w}}_i(k)\|\|\mathbf{w}_i\|}, \quad (58)$$

where $\widetilde{\mathbf{w}}_i$ and $\mathbf{w}_i$ are the ith estimated and exact generalized eigenvectors, respectively. Here the result of (58) is averaged over 100 independent trials. Moreover, to measure the numerical stability of considered algorithms, we make use of the sample standard deviation (SSD) of the direction cosine defined as:

$$\mathrm{SSD}_i(k) = \sqrt{\frac{1}{L-1}\sum_{j=1}^{L}\left[\mathrm{DC}_{i,j}(k) - \overline{\mathrm{DC}_i}(k)\right]^2}, \quad (59)$$

where $\mathrm{DC}_{i,j}(k)$ is the direction cosine of jth independent trial, where $j = 1, \cdots, L$, of the ith estimated generalized eigenvector, and $\overline{\mathrm{DC}_i}(k)$ is the direction cosine of ith estimated generalized eigenvector averaged over L trials. Here $L = 100$.

The results for the first two generalized eigenvectors are displayed in Figs 1. We make several observations: i) for the first generalized eigenvector, Algorithm 2 outperforms other algorithms in terms of convergence speed, estimation accuracy and numerical stability. ii) For the second generalized eigenvector, again Algorithm 2 shows the best performance among the algorithms considered in convergence speed and numerical stability, while its estimation ac-

curacy being similar to that of the gradient-based method. iii) Algorithm 1 shows similar performance as the PI-based method, however it is computationally more efficient (see Section 3.2). iv) The gradient-based method has the slowest convergence speed among all methods.

5. CONCLUSION

In this paper, we proposed the APR-EVD algorithm for standard eigenvalue decomposition through randomization. By exploiting rank-1 update strategy, we developed two online algorithms based on APR-EVD for generalized eigenvectors extraction. Our numerical results show that Algorithm 2 outperforms the compared algorithms in tracking the first two dominant generalized eigenvectors in terms of convergence speed, estimation accuracy and numerical stability. Further, although Algorithm 1 has similar performance as the PI-based method, it has lower computational cost.

6. REFERENCES

[1] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, *Templates for the solution of Algebraic Eigenvalue Problems: A Practical Guide*, SIAM, (2000).

[2] L. Parra and P. Sajda, " Blind source separation via generalized eigenvalue decomposition," *Journal of Machine Learning Research*, vol. 4, no. 4, pp. 1261–1269, Dec 2003.

[3] G. Yingbin, K. Xiangyu, Z. Zhengxin, and H. Li'an, "An adaptive self-stabilizing algorithm for minor generalized eigenvector extraction and its convergence analysis," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4869–4881, Oct 2018.

[4] X. Han and L. Clemmensen, "Regularized generalized eigendecomposition with applications to sparse supervised feature extraction and sparse discriminant analysis," *Pattern Recognition*, vol. 49, pp. 43–54, Dec 2016.

[5] G. Yuan, L. Shen, and W. Zheng, "A Decomposition Algorithm for the Sparse Generalized Eigenvalue Problem," in *CVPR*, 2019, pp. 6113–6122.

[6] A. Valizadeh and M. Najibi, " A constrained optimization approach for an adaptive generalized subspace tracking algorithm," *Computers &Electrical Engineering*, vol. 36, no. 4, pp. 596–602, 2010.

[7] H. Chen, G. Jiang, and K. Yoshihira, "Failure detection in large-scale internet services by principal subspace mapping," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 10, pp. 1308–1320, 2007.

[8] D. R. Morgan, "Adaptive algorithms for solving generalized eigenvalue signal enhancement problems," *Signal processing*, vol. 84, no. 6, pp. 957–968, 2004.

[9] S. Ding, X. Hua, and J. Yu, "An overview on nonparallel hyperplane support vector machine algorithms," *Neural computing and applications*, vol. 25, no. 5, pp. 975–982, 2014.

[10] J. Benesty, M. G. Christensen, and J. R. Jensen, *Signal enhancement with variable span linear filters*. Springer, 2016, vol. 7.

[11] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed., Johns Hopkins Univ. Press, Baltimore, MD, (1996).

[12] S. Choi, J. Choi, H.-J. Im, and B. Choi, "A novel adaptive beamforming algorithm for antenna array cdma systems with strong interferers," *IEEE Transactions on Vehicular Technology*, vol. 51, no. 5, pp. 808–816, 2002.

[13] R. Moller and A. Konies, "Coupled principal component analysis," *IEEE Transactions on Neural Networks*, vol. 15, no. 1, pp. 214–222, 2004.

[14] X. Feng, X. Kong, Z. Duan, and H. Ma, "Adaptive generalized eigen-pairs extraction algorithms and their convergence analysis," *IEEE Transactions on Signal Processing*, vol. 64, no. 11, pp. 2976–2989, 2016.

[15] T. D. Nguyen, N. Takahashi, and I. Yamada, "An adaptive extraction of generalized eigensubspace by using exact nested orthogonal complement structure," *Multidimensional Systems and Signal Processing*, vol. 24, no. 3, pp. 457–483, 2013.

[16] J. Yang, H. Xi, F. Yang, and Y. Zhao, "RLS-based adaptive algorithms for generalized eigen-decomposition," *IEEE Transactions on Signal Processing*, vol. 54, no. 4, pp. 1177–1188, 2006.

[17] B. Yang, "Projection approximation subspace tracking," *IEEE Transactions on Signal processing*, vol. 43, no. 1, pp. 95–107, 1995.

[18] S. Attallah and K. Abed-Meraim, "A fast adaptive algorithm for the generalized symmetric eigenvalue problem," *IEEE Signal Processing Letters*, vol. 15, pp. 797–800, 2008.

[19] T. Tanaka, "Fast generalized eigenvector tracking based on the power method," *IEEE Signal Processing Letters*, vol. 16, no. 11, pp. 969–972, 2009.

[20] N. Halko, P.-G. Martinsson, and J. Tropp, "Finding Structure with Randomness: Probabilistic Algorithms for Constructing Approximate Matrix Decompositions," *SIAM Review*, vol. 53, no. 2, pp. 217–288, Jun 2011.

[21] M. Gu, "Subspace Iteration Randomization and Singular Value Problems," *SIAM J. Sci. Comput.*, vol. 37, no. 3, pp. A1139–A1173, 2015.

[22] M. F. Kaloorazi and J. Chen, "Randomized Truncated Pivoted QLP Factorization for Low-Rank Matrix Recovery," *IEEE Signal Processing Letters*, vol. 26, no. 7, pp. 1075–1079, Jul 2019.

[23] T. D. Nguyen and I. Yamada, "Adaptive normalized quasi-newton algorithms for extraction of generalized eigen-pairs and their convergence analysis," *IEEE Transactions on Signal Processing*, vol. 61, no. 6, pp. 1404–1418, 2012.

[24] G. W. Stewart, *Matrix Algorithms: Volume 1: Basic Decompositions*, SIAM, Philadelphia, PA, (1998).

[25] Y. Saad, *Numerical methods for large eigenvalue problems*. SIAM, 2nd Ed., 2011.

[26] G. W. Stewart, "A generalization of saad's theorem on rayleigh–ritz approximations," *Linear Algebra and its Applications*, vol. 327, no. 1-3, pp. 115–119, 2001.