Chapter 8
Coupled Principal Component Analysis

8.1 Introduction

Among neural network-based PCA or MCA algorithms, most previously reviewed do not consider eigenvalue estimates in the update equations of the weights, except an attempt to control the learning rate based on the eigenvalue estimates [1]. In [2], Moller provided a framework for a special class of learning rules where eigenvectors and eigenvalues are simultaneously estimated in coupled update equations, and has proved that coupled learning algorithms are solutions for the speed stability problem that plagues most noncoupled learning algorithms. The convergence speed of a system depends on the eigenvalues of its Jacobian, which vary with the eigenvalues of the covariance matrix in noncoupled PCA/MCA algorithms [2]. Moller showed that, in noncoupled PCA algorithms, the eigen motion in all directions mainly depends on the principal eigenvalue of the covariance matrix [2]. Numerical stability and fast convergence of algorithms can only be achieved by guessing this eigenvalue in advance [2]. In particular for chains of principal component analyzers which simultaneously estimate the first few principal eigenvectors [3], choosing the right learning rates for all stages may be difficult. The problem is even more severe for MCA algorithms. MCA algorithms exhibit a wide range of convergence speeds in different eigen directions, since the eigenvalues of the Jacobian cover approximately the same range as the eigenvalues of the covariance matrix. Using small enough learning rates to still guarantee the stability of the numerical procedure, noncoupled MCA algorithms may converge very slowly [2].

In [2], Moller derived a coupled learning rule by applying Newton's method to a common information criterion. A Newton descent yields learning rules with approximately equal convergence speeds in all eigen directions of the system. Moreover, all eigenvalues of the Jacobian of such a system are approximately. Thus, the dependence on the eigenvalues of the covariance matrix can be eliminated [2]. Moller showed that with respect to averaged differential equations, this

X. Kong et al., *Principal Component Analysis Networks and Algorithms*,
DOI 10.1007/978-981-10-2915-8_8

approach solves the speed stability problem for both PCA and MCA rules. However, these differential equations can only be turned into the aforementioned online rules for the PCA but not for the MCA case, leaving the more severe MCA stability problem still unresolved [2]. Interestingly, unlike most existing adaptive algorithms, the coupled learning rule for the HEP effectively utilizes the latest estimate of the eigenvalue to update the estimate of the eigenvector [4]. Numerical examples in [2] showed that this algorithm achieves fast and stable convergence for both low-dimensional data and high-dimensional data. Unfortunately, there has been no report about any explicit convergence analysis for the coupled learning rule. Thus, the condition for the convergence to the desired eigen pair is not clear; e.g., the region within which the initial estimate of the eigen pair must be chosen to guarantee the convergence to the desired eigen pair has not yet been known [4].

Recently, Tuan Duong Nguyen et al. proposed novel algorithms in [4] for given explicit knowledge of the matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$. These algorithms for estimating the generalized eigen pair associated with the largest/smallest generalized eigenvalue are designed (i) based on a new characterization of the generalized eigen pair as a stationary point of a certain function and (ii) by combining a normalization step and quasi-Newton step at each update. Moreover, the rigorous convergence analysis of the algorithms was established by the DDT approach. For adaptive implementation of the algorithms, Tuan Duong Nguyen et al. proposed to use the exponentially weighted sample covariance matrices and the Sherman–Morrison–Woodbury matrix-inversion lemma.

The aim of this chapter was to develop some coupled PCA or coupled generalized PCA algorithms. First, on the basis of a special information criterion in [5], we propose a coupled dynamical system by modifying Newton's method in this chapter. Based on the coupled system and some approximation, we derive two CMCA algorithms and two CPCA algorithms; thus, two unified coupled algorithms are obtained [6]. Then, we propose a coupled generalized system in this chapter, which is obtained by using the Newton's method and a novel generalized information criterion. Based on this coupled generalized system, we obtain two coupled algorithms with normalization steps for minor/principal generalized eigen pair extraction. The technique of multiple generalized eigen pair extraction is also introduced in this chapter. The convergence of algorithms is justified by DDT system.

In this chapter, we will review and discuss the existing coupled PCA or coupled generalized PCA algorithms. Two coupled algorithms proposed by us will be analyzed in detail. The remainder of this chapter is organized as follows. An overview of the existing coupled PCA or coupled generalized PCA algorithms is presented in Sect. 8.2. An unified and coupled self-stabilizing algorithm for minor and principal eigen pair extraction algorithms are discussed in Sect. 8.3. An adaptive generalized eigen pair extraction algorithms and their convergence analysis via DDT method are presented in Sect. 8.4, followed by summary in Sect. 8.5.

8.2 Review of Coupled Principal Component Analysis

8.2.1 Moller's Coupled PCA Algorithm

Learning rules for principal component analysis are often derived by optimizing some information criterion, e.g., by maximizing the variance of the projected data or by minimizing the reconstruction error [2, 7]. In [2], Moller proposed the following information criterion as the starting point of his analysis

$$p = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w}\lambda^{-1} - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + \ln\lambda. \tag{8.1}$$

where $\boldsymbol{w}$ denotes an n-dimensional weight vector, i.e., the estimate of the eigenvector, λ is the eigenvalue estimate, and $\boldsymbol{C} = E\{\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\}$ is the $n \times n$ covariance matrix of the data. From (8.1), by using the gradient method and the Newton descent, Moller derived a coupled system of differential equations for the PCA case

$$\dot{\boldsymbol{w}} = \boldsymbol{C}\boldsymbol{w}\lambda^{-1} - \boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w}\lambda^{-1} - \frac{1}{2}\boldsymbol{w}\,(1 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}), \tag{8.2}$$

$$\dot{\lambda} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w} - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}\lambda, \tag{8.3}$$

and another for MCA case

$$\dot{\boldsymbol{w}} = \boldsymbol{C}^{-1}\boldsymbol{w}\lambda + \boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w}\lambda^{-1} - \frac{1}{2}\boldsymbol{w}\,(1 + 3\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}), \tag{8.4}$$

$$\dot{\lambda} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w} - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}\lambda. \tag{8.5}$$

For the stability of the above algorithms, see [2]. It has been shown that for the above coupled PCA system, if we assume $\lambda_j \ll \lambda_1$, the system converges with approximately equal speeds in all its eigen directions, and this speed is widely independent of the eigenvalues λ_j of the covariance matrix. And for the above coupled MCA system, if we assume $\lambda_1 \ll \lambda_j$, then the convergence speed is again about equal in all eigen directions and independent of the eigenvalues of $\boldsymbol{C}$.

By informally approximating $\boldsymbol{C} \approx \boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}$, the averaged differential equations of (8.2) and (8.3) can be turned into an online learning rule:

$$\dot{\boldsymbol{w}} = \gamma\left[y\lambda^{-1}(\boldsymbol{x} - \boldsymbol{w}y) - \frac{1}{2}\boldsymbol{w}\,(1 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w})\right], \tag{8.6}$$

$$\dot{\lambda} = \gamma(y^2 - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}\lambda). \tag{8.7}$$

According to the stochastic approximation theory, the resulting stochastic differential equation has the same convergence goal as the deterministic averaged equation if certain conditions are fulfilled, the most important of which is that a learning rate decreases to zero over time. The online rules (8.6) and (8.7) can be understood as a

learning rule for the weight vector $\boldsymbol{w}$ of a linear neuron which computes its output y from the scalar product of weight vector and input vector $\boldsymbol{y} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$.

In [2], the analysis of the temporal derivative of the (squared) weight vector length in (8.6) has shown that the weight vector length may in general be fluctuating. By further approximating $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} \approx 1$ (which is fulfilled in the vicinity of the stationary points) in the averaged systems (8.2) and (8.3), the following system can be derived

$$\dot{\boldsymbol{w}} = \boldsymbol{C}\boldsymbol{w}\lambda^{-1} - \boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w}\lambda^{-1}, \tag{8.8}$$

$$\dot{\lambda} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w} - \lambda. \tag{8.9}$$

This learning rule system is known as ALA [1]. The eigenvalues of the system's Jacobian are still approximately equal and widely independent of the eigenvalues of the covariance matrix. The corresponding online system is given by

$$\dot{\boldsymbol{w}} = \gamma y\lambda^{-1}(\boldsymbol{x} - \boldsymbol{w}y), \tag{8.10}$$

$$\dot{\lambda} = \gamma(y^2 - \lambda). \tag{8.11}$$

It is obvious that ALA can be interpreted as an instance of Oja's PCA rule.

From (8.4) and (8.5), it has been shown that having a Jacobian with eigenvalues that are equal and widely independent of the eigenvalues of the covariance matrix appears to be a solution for the speed stability problem. However, when attempting to turn this system into an online rule, a problem is encountered when replacing the inverse covariance matrix $\boldsymbol{C}^{-1}$ by a quantity including the input vector $\boldsymbol{x}$. An averaged equation linearly depending on C takes the form $\dot{\boldsymbol{w}} = f(\boldsymbol{C}, \boldsymbol{w}) = f(E\{\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\}, \boldsymbol{w}) = E\{f(\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}, \boldsymbol{w})\}$. In an online rule, the expectation of the gradient is approximated by slowly following $\dot{\boldsymbol{w}} = \gamma f(\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}, \boldsymbol{w})$ for subsequent observations of $\boldsymbol{x}$. This transition is obviously not possible if the equation contains $\boldsymbol{C}^{-1}$. Thus, there is no online version for the MCA systems (8.4) and (8.5). Despite using the ALA-style normalization, the convergence speed in different eigen directions still depends on the entire range of eigenvalues of the covariance matrix. So the speed stability problem still exists.

8.2.2 *Nguyen's Coupled Generalized Eigen pairs Extraction Algorithm*

In [8], Nguyen proposed a generalized principal component analysis algorithm and its differential equation form is given as:

$$\dot{\boldsymbol{w}} = \boldsymbol{R}_x^{-1}\boldsymbol{R}_y\boldsymbol{w} - \boldsymbol{w}^H\boldsymbol{R}_y\boldsymbol{w}\boldsymbol{w}. \tag{8.12}$$

Let $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N]$, in which $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N$ are the generalized eigenvectors of matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$. The Jacobian in the stationary point is given as:

$$\boldsymbol{J}(\boldsymbol{w}_1) = \frac{\partial \dot{\boldsymbol{w}}}{\partial \boldsymbol{w}^{\mathrm{T}}}|_{\boldsymbol{w}=\boldsymbol{w}_1} = \boldsymbol{R}_x^{-1}\boldsymbol{R}_y - \lambda_1 \boldsymbol{I} - 2\lambda_1 \boldsymbol{w}_1 \boldsymbol{w}_1^H \boldsymbol{R}_x. \tag{8.13}$$

Solving for the eigenvectors of $\boldsymbol{J}$ can be simplified to the solving for the eigenvector of its similar diagonally matrix $\boldsymbol{J}^* = \boldsymbol{P}^{-1}\boldsymbol{J}\boldsymbol{P}$, since $\boldsymbol{J}^*$ and $\boldsymbol{J}$ have the same eigenvectors and eigenvalues, and the eigenvalues of diagonal matrix $\boldsymbol{J}^*$ are easy to be obtained. Considering $\boldsymbol{W}^H \boldsymbol{R}_x \boldsymbol{W} = \boldsymbol{I}$, let $\boldsymbol{P} = \boldsymbol{W}$. Then we have $\boldsymbol{P}^{-1} = \boldsymbol{W}^H \boldsymbol{R}_x$. Thus, it holds that

$$\begin{aligned} \boldsymbol{J}^*(\boldsymbol{w}_1) &= \boldsymbol{W}^H \boldsymbol{R}_x \left(\boldsymbol{R}_x^{-1}\boldsymbol{R}_y - \lambda_1 \boldsymbol{I} - 2\lambda_1 \boldsymbol{w}_1 \boldsymbol{w}_1^H \boldsymbol{R}_x\right) \boldsymbol{W} \\ &= \boldsymbol{\Lambda} - \lambda_1 \boldsymbol{I} - 2\lambda_1 \boldsymbol{W}^H \boldsymbol{R}_x \boldsymbol{w}_1 \left(\boldsymbol{W}^H \boldsymbol{R}_x \boldsymbol{w}_1\right)^H \frac{\Delta y}{\Delta x}. \end{aligned} \tag{8.14}$$

Since $\boldsymbol{W}^H \boldsymbol{R}_x \boldsymbol{w}_1 = \boldsymbol{e}_1 = [1, 0, \ldots, 0]^{\mathrm{T}}$, (8.14) will be reduced to

$$\boldsymbol{J}^*(\boldsymbol{w}_1) = \boldsymbol{\Lambda} - \lambda_1 \boldsymbol{I} - 2\lambda_1 \boldsymbol{e}_1 \boldsymbol{e}_1^H. \tag{8.15}$$

The eigenvalues α determined from $\det(\boldsymbol{J}^* - \alpha \boldsymbol{I}) = 0$ are given as:

$$\alpha_1 = -2\lambda_1, \ \alpha_j = \lambda_j - \lambda_1, \ j = 2, \ldots, N. \tag{8.16}$$

Since the stability requires $\alpha < 0$ and thus $\lambda_1 \gg \lambda_j$, $j = 2, 3, \ldots, n$, it can be seen that only principal eigenvector–eigenvalue pairs are stable stationary points, and all other stationary points are saddles or repellers, which can still testify that (8.12) is a generalized PCA algorithm. In the practical signal processing applications, it always holds that $\lambda_1 \gg \lambda_j$, $j = 2, 3, \ldots, n$. Thus, $\alpha_j \approx -\lambda_1$, i.e., the eigen motion in all directions in algorithm (8.12) depends on the principal eigenvalue of the covariance matrix. Thus, this algorithm has the speed stability problem.

In [4], an adaptive normalized quasi-Newton algorithm for generalized eigen pair extraction was proposed and its convergence analysis was conducted. This algorithm is a coupled generalized eigen pair extraction algorithm, which can be interpreted as natural combinations of the normalization step and quasi-Newton steps for finding the stationary points of the function

$$\xi(\boldsymbol{w}, \lambda) = \boldsymbol{w}^H \boldsymbol{R}_y \boldsymbol{w} \lambda^{-1} - \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} + \ln \lambda, \tag{8.17}$$

which is a generalization of the information criterion introduced in [2] for the HEP. The stationary point of ξ is defined as a zero of

$$\begin{pmatrix} \frac{\partial \xi}{\partial \boldsymbol{w}} \\ \frac{\partial \xi}{\partial \lambda} \end{pmatrix} = \begin{pmatrix} 2\boldsymbol{R}_y \boldsymbol{w} \lambda^{-1} - 2\boldsymbol{R}_x \boldsymbol{w} \\ -\boldsymbol{w}^H \boldsymbol{R}_y \boldsymbol{w} \lambda^{-2} + \lambda^{-1} \end{pmatrix}. \tag{8.18}$$

Hence, from

$$\begin{cases} \boldsymbol{R}_y\bar{\boldsymbol{w}} = \bar{\lambda}\boldsymbol{R}_x\bar{\boldsymbol{w}} \\ \bar{\boldsymbol{w}}^H\boldsymbol{R}_y\bar{\boldsymbol{w}} = \bar{\lambda} \end{cases}, \tag{8.19}$$

it can be seen that $(\bar{\boldsymbol{w}}, \bar{\lambda}) \in C^N \times \Re$ is a stationary point of ξ, which implies that a stationary point $(\bar{\boldsymbol{w}}, \bar{\lambda})$ of ξ is a generalized eigen pair of the matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$. To avoid these computational difficulties encountered in Newton's method, Nguyen proposed to use approximations of $\boldsymbol{H}^{-1}(\boldsymbol{w}, \lambda)$ in the vicinity of two stationary points of their special interest

$$\boldsymbol{H}^{-1}(\boldsymbol{w}, \lambda) \approx \widetilde{\boldsymbol{H}}_P^{-1}(\boldsymbol{w}, \lambda) = \frac{1}{2}\begin{pmatrix} \frac{1}{2}\boldsymbol{w}\boldsymbol{w}^H - \boldsymbol{R}_x^{-1} & -\boldsymbol{w}\lambda \\ -\boldsymbol{w}^H\lambda & 0 \end{pmatrix}, \tag{8.20}$$

for $(\boldsymbol{w}, \lambda) \approx (\boldsymbol{v}_N, \lambda_N)$, and

$$\boldsymbol{H}^{-1}(\boldsymbol{w}, \lambda) \approx \widetilde{\boldsymbol{H}}_M^{-1}(\boldsymbol{w}, \lambda) = \frac{1}{2}\begin{pmatrix} \boldsymbol{R}_y^{-1}\lambda - \frac{3}{2}\boldsymbol{w}\boldsymbol{w}^H & -\boldsymbol{w}\lambda \\ -\boldsymbol{w}^H\lambda & 0 \end{pmatrix}, \tag{8.21}$$

for $(\boldsymbol{w}, \lambda) \approx (\boldsymbol{v}_1, \lambda_1)$. By applying Newton's strategy for finding the stationary point of ξ using the gradient (8.18) and the approximations (8.20) and (8.21), a learning rule for estimating the generalized eigen pair associated with the largest generalized eigenvalue was obtained as:

$$\begin{aligned} \boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \eta_1\{&\boldsymbol{R}_x^{-1}\boldsymbol{R}_y\boldsymbol{w}(k)\lambda^{-1}(k) \\ &-\boldsymbol{w}^H(k)\boldsymbol{R}_y\boldsymbol{w}(k)\,\boldsymbol{w}(k)\lambda^{-1}(k) - \frac{1}{2}\boldsymbol{w}(k)[1 - \boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{w}(k)]\Big\}, \end{aligned} \tag{8.22}$$

$$\lambda(k+1) = \lambda(k) + \gamma_1\left[\boldsymbol{w}^H(k+1)\boldsymbol{R}_y\boldsymbol{w}(k+1) - \boldsymbol{w}^H(k+1)\boldsymbol{R}_x\boldsymbol{w}(k+1)\lambda(k)\right], \tag{8.23}$$

and a learning rule for estimating the generalized eigen pair associated with the smallest generalized eigenvalue was obtained as:

$$\begin{aligned} \boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \eta_2\Big\{&\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)\lambda(k) + \boldsymbol{w}^H(k)\boldsymbol{R}_y\boldsymbol{w}(k)\,\boldsymbol{w}(k)\lambda^{-1}(k) \\ &-\frac{1}{2}\boldsymbol{w}(k)\left[1 + 3\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{w}(k)\right]\Big\}, \end{aligned} \tag{8.24}$$

$$\lambda(k+1) = \lambda(k) + \gamma_2\left[\boldsymbol{w}^H(k+1)\boldsymbol{R}_y\boldsymbol{w}(k+1) - \boldsymbol{w}^H(k+1)\boldsymbol{R}_x\boldsymbol{w}(k+1)\lambda(k)\right], \tag{8.25}$$

where $\eta_1, \gamma_1, \eta_2, \gamma_2 > 0$ are the step sizes, and $[\boldsymbol{w}(k), \lambda(k)]$ is the estimate at time k of the generalized eigen pair associated with the largest/smallest generalized eigenvalue. By introducing the normalization step in the above learning rules at each update, using the exponentially weighted sample covariance matrices $\widehat{\boldsymbol{R}}_y$ and

$\widehat{\boldsymbol{R}}_{\boldsymbol{x}}$ which are updated recursively, and using the Sherman–Morrison–Woodbury matrix-inversion lemma, Nguyen's coupled generalized eigen pair extraction algorithm was obtained as follows.

Adaptive coupled generalized PCA algorithm:

$$\begin{aligned}\tilde{\boldsymbol{w}}(k) = \boldsymbol{w}(k-1) + \frac{\eta_1}{\lambda(k-1)}\Big(&\boldsymbol{Q}_{\boldsymbol{x}}(k)\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k)\boldsymbol{w}(k-1) \\ &-\boldsymbol{w}^H(k-1)\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k)\boldsymbol{w}(k-1)\boldsymbol{w}(k-1)\Big),\end{aligned} \tag{8.26}$$

$$\boldsymbol{w}(k) = \frac{\tilde{\boldsymbol{w}}(k)}{\|\tilde{\boldsymbol{w}}(k)\|_{\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k)}}, \tag{8.27}$$

$$\lambda(k) = (1-\gamma_1)\lambda(k-1) + \gamma_1 \boldsymbol{w}^H(k)\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k)\boldsymbol{w}(k), \tag{8.28}$$

and adaptive coupled generalized MCA algorithm:

$$\begin{aligned}\tilde{\boldsymbol{w}}(k) = \boldsymbol{w}(k-1) + \eta_2\Big(&\boldsymbol{Q}_{\boldsymbol{y}}(k)\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k)\boldsymbol{w}(k-1)\lambda(k-1) \\ &+\boldsymbol{w}^H(k-1)\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k)\,\boldsymbol{w}(k-1)\boldsymbol{w}(k-1)\lambda^{-1}(k-1) - 2\boldsymbol{w}(k-1)\Big),\end{aligned} \tag{8.29}$$

$$\boldsymbol{w}(k) = \frac{\tilde{\boldsymbol{w}}(k)}{\|\tilde{\boldsymbol{w}}(k)\|_{\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k)}}, \tag{8.30}$$

$$\lambda(k) = (1-\gamma_2)\lambda(k-1) + \gamma_2 \boldsymbol{w}^H(k)\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k)\boldsymbol{w}\,(k), \tag{8.31}$$

where $\|\boldsymbol{u}\|_{\boldsymbol{R}_{\boldsymbol{x}}} = \sqrt{\boldsymbol{u}^H\boldsymbol{R}_{\boldsymbol{x}}\boldsymbol{u}}$ is defined as the $\boldsymbol{R}_{\boldsymbol{x}}$-norm, $\boldsymbol{Q}_{\boldsymbol{x}} = \boldsymbol{R}_{\boldsymbol{x}}^{-1}$, $\boldsymbol{Q}_{\boldsymbol{y}} = \boldsymbol{R}_{\boldsymbol{y}}^{-1}$, which are updated recursively as follows:

$$\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k+1) = \beta\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k) + \boldsymbol{y}(k+1)\boldsymbol{y}^H(k+1), \tag{8.32}$$

$$\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k+1) = \alpha\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k) + \boldsymbol{x}(k+1)\boldsymbol{x}^H(k+1), \tag{8.33}$$

$$\boldsymbol{Q}_{\boldsymbol{x}}(k+1) = \frac{1}{\alpha}\left(\boldsymbol{Q}_{\boldsymbol{x}}(k) - \frac{\boldsymbol{Q}_{\boldsymbol{x}}(k)\boldsymbol{x}(k+1)\boldsymbol{x}^H(k+1)\boldsymbol{Q}_{\boldsymbol{x}}(k)}{\alpha + \boldsymbol{x}^H(k+1)\boldsymbol{Q}_{\boldsymbol{x}}(k)\boldsymbol{x}(k+1)}\right), \tag{8.34}$$

$$\boldsymbol{Q}_{\boldsymbol{y}}(k+1) = \frac{1}{\beta}\left(\boldsymbol{Q}_{\boldsymbol{y}}(k) - \frac{\boldsymbol{Q}_{\boldsymbol{y}}(k)\boldsymbol{y}(k+1)\boldsymbol{y}^H(k+1)\boldsymbol{Q}_{\boldsymbol{y}}(k)}{\beta + \boldsymbol{y}^H(k+1)\boldsymbol{Q}_{\boldsymbol{y}}(k)\boldsymbol{y}(k+1)}\right), \tag{8.35}$$

where $\alpha, \beta \in (0,1)$ are the forgetting factors.

Different from the analysis of Möller's coupled algorithm, the convergence analysis of Nguyen algorithm in [4] was not conducted via the eigenvalue of Jacobian matrix. Nguyen established rigorous analysis of the DDT systems

showing that, for a step size within a certain range, the algorithm converges to the orthogonal projection of the initial estimate onto the generalized eigen-subspace associated with the largest/smallest generalized eigenvalue.

Next, we analyze the convergence of Nguyen's algorithm via the eigenvalues of Jacobian matrix.

By ignoring the normalization step (8.27), the differential equation form of GPCA algorithms (8.26) and (8.28) can be written as:

$$\dot{\boldsymbol{w}} = \lambda^{-1}\left(\boldsymbol{R}_x^{-1}\boldsymbol{R}_y\boldsymbol{w} - \boldsymbol{w}^H\boldsymbol{R}_y\boldsymbol{w}\boldsymbol{w}\right), \tag{8.36}$$

$$\dot{\lambda} = \boldsymbol{w}^H\boldsymbol{R}_y\boldsymbol{w} - \lambda. \tag{8.37}$$

The Jacobian matrix at the stationary point $(\boldsymbol{w}_1, \lambda_1)$ is given as:

$$\begin{aligned}\boldsymbol{J}(\boldsymbol{w}_1, \lambda_1) &= \left.\begin{pmatrix} \frac{\partial \dot{\boldsymbol{w}}}{\partial \boldsymbol{w}^{\mathrm{T}}} & \frac{\partial \dot{\boldsymbol{w}}}{\partial \lambda} \\ \frac{\partial \dot{\lambda}}{\partial \boldsymbol{w}^{\mathrm{T}}} & \frac{\partial \dot{\lambda}}{\partial \lambda} \end{pmatrix}\right|_{(\boldsymbol{w}_1, \lambda_1)}, \\ &= \begin{pmatrix} \lambda_1^{-1}\boldsymbol{R}_x^{-1}\boldsymbol{R}_y - \boldsymbol{I} - 2\boldsymbol{w}_1\boldsymbol{w}_1^H\boldsymbol{R}_x & \boldsymbol{0} \\ 2\lambda_1\boldsymbol{w}_1^H\boldsymbol{R}_x & -1 \end{pmatrix}.\end{aligned} \tag{8.38}$$

Let

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{W} & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{pmatrix}. \tag{8.39}$$

Then, it can be easily seen that

$$\boldsymbol{P}^{-1} = \begin{pmatrix} \boldsymbol{W}^H\boldsymbol{R}_x & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{pmatrix}. \tag{8.40}$$

Solving for the eigenvectors of $\boldsymbol{J}$ can then be simplified to the solving for the eigenvector of its similar diagonally matrix $\boldsymbol{J}^* = \boldsymbol{P}^{-1}\boldsymbol{J}\boldsymbol{P}$. Then it holds that

$$\boldsymbol{J}^*(\boldsymbol{w}_1, \lambda_1) = \begin{pmatrix} \lambda_1^{-1}\boldsymbol{\Lambda} - \boldsymbol{I} - 2\boldsymbol{e}_1\boldsymbol{e}_1^H & \boldsymbol{0} \\ 2\lambda_1\boldsymbol{e}_1^H & -1 \end{pmatrix}. \tag{8.41}$$

The eigenvalues α determined from $\det(\boldsymbol{J}^* - \alpha\boldsymbol{I}) = 0$ are

$$\alpha_1 = -2,\ \alpha_{N+1} = -1,\ \alpha_j = \frac{\lambda_j}{\lambda_1} - 1,\ j = 2, \ldots, N. \tag{8.42}$$

Since the stability requires $\alpha < 0$ and thus $\lambda_j < \lambda_1,\ j = 2, 3, \ldots, n$, it can be seen that only principal eigenvector–eigenvalue pairs are stable stationary points, and all other stationary points are saddles or repellers. If we further assume that $\lambda_1 \gg \lambda_j$,

then $\alpha_j \approx -1, j = 2, 3, \ldots, n$. That is to say, the eigen motion in all directions in the algorithm do not depend on the generalized eigenvalue of the covariance matrix of input signal. Thus, this algorithm does not have the speed stability problem. Similar analysis can be applied to the GMCA algorithms (8.29) and (8.31).

8.2.3 *Coupled Singular Value Decomposition of a Cross-Covariance Matrix*

In [9], a coupled online learning rule for the singular value decomposition (SVD) of a cross-covariance matrix was derived. In coupled SVD rules, the singular value is estimated alongside the singular vectors, and the effective learning rates for the singular vector rules are influenced by the singular value estimates [9]. In addition, a first-order approximation of Gram–Schmidt orthonormalization as decorrelation method for the estimation of multiple singular vectors and singular values was used. It has been shown that the coupled learning rules converge faster than Hebbian learning rules and that the first-order approximation of Gram–Schmidt orthonormalization produces more precise estimates and better orthonormality than the standard deflation method [9].

The neural network and its learning algorithm for the singular value decomposition of a cross-covariance matrix will be discussed in Chap. 9, in which the coupled online learning rules for the SVD of a cross-covariance matrix will be analyzed in detail.

8.3 Unified and Coupled Algorithm for Minor and Principal Eigen Pair Extraction

Coupled algorithm can mitigate the speed stability problem which exists in most noncoupled algorithms. Though unified algorithm and coupled algorithm have these advantages over single purpose algorithm and noncoupled algorithm, respectively, there are only few of unified algorithms, and coupled algorithms have been proposed. Moreover, to the best of the authors' knowledge, there are no both unified and coupled algorithms which have been proposed. In this chapter, based on a novel information criterion, we propose two self-stabilizing algorithms which are both unified and coupled. In the derivation of our algorithms, it is easier to obtain the results compared with traditional methods, because there is no need to calculate the inverse Hessian matrix. Experiment results show that the proposed algorithms perform better than existing coupled algorithms and unified algorithms.

8.3.1 Couple Dynamical System

The derivation of neural network learning rules often starts with an information criterion, e.g., by maximization of the variance of the projected data or by minimization of the reconstruction error [7]. However, as stated in [10], the freedom of choosing an information criterion is greater if Newton's method is applied because the criterion just has to have stationary points in the desired solutions. Thus in [2], Moller proposed a special criterion. Based on this criterion and by using Newton's method, Moller derived some CPCA learning rules and a CMCA learning rule. Based on another criterion, Hou [5] derived the same CPCA and CMCA learning rules as that of Moller's, and Appendix 2 of [5] showed that it is easier and clearer to approximate the inverse of the Hessian.

To start the analysis, we use the same information criterion as Hou's, which is

$$p = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w} - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}\lambda + \lambda \tag{8.43}$$

where $\boldsymbol{C} = E\{\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\} \in \Re^{n\times n}$ is the covariance matrix of the n-dimensional input data sequence $\boldsymbol{x}$, $\boldsymbol{w} \in \Re^{n\times 1}$ and $\lambda \in \Re$ denotes the estimation of eigenvector (weight vector) and eigenvalue of $\boldsymbol{C}$, respectively.

It is found that

$$\frac{\partial p}{\partial \boldsymbol{w}} = 2\boldsymbol{C}\boldsymbol{w} - 2\lambda\boldsymbol{w} \tag{8.44}$$

$$\frac{\partial p}{\partial \lambda} = -\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} + 1. \tag{8.45}$$

Thus, the stationary points $(\bar{\boldsymbol{w}}, \bar{\lambda})$ of (8.43) are defined by

$$\left.\frac{\partial p}{\partial \boldsymbol{w}}\right|_{(\bar{\boldsymbol{w}},\bar{\lambda})} = \boldsymbol{0},\ \left.\frac{\partial p}{\partial \lambda}\right|_{(\bar{\boldsymbol{w}},\bar{\lambda})} = 0. \tag{8.46}$$

Then, we can obtain

$$\boldsymbol{C}\bar{\boldsymbol{w}} = \bar{\lambda}\bar{\boldsymbol{w}}, \tag{8.47}$$

$$\bar{\boldsymbol{w}}^{\mathrm{T}}\bar{\boldsymbol{w}} = 1 \tag{8.48}$$

from which we can also conclude that $\bar{\boldsymbol{w}}^{\mathrm{T}}\boldsymbol{C}\bar{\boldsymbol{w}} = \bar{\lambda}$. Thus, the criterion (8.43) fulfills the aforementioned requirement: The stationary points include all associated eigenvectors and eigenvalues of $\boldsymbol{C}$. The Hessian of the criterion is given as:

$$\boldsymbol{H}(\boldsymbol{w},\lambda)=\begin{pmatrix}\frac{\partial^2 p}{\partial \boldsymbol{w}^2} & \frac{\partial^2 p}{\partial \boldsymbol{w}\partial\lambda}\\ \frac{\partial^2 p}{\partial\lambda\partial \boldsymbol{w}} & \frac{\partial^2 p}{\partial\lambda^2}\end{pmatrix}=2\begin{pmatrix}\boldsymbol{C}-\lambda\boldsymbol{I} & -\boldsymbol{w}\\ -\boldsymbol{w}^{\mathrm{T}} & 0\end{pmatrix}. \tag{8.49}$$

Based on the Newton's method, the equation used by Moller and Hou to derive the differential equations can be written as:

$$\begin{pmatrix}\dot{\boldsymbol{w}}\\ \dot{\lambda}\end{pmatrix}=-\boldsymbol{H}^{-1}(\boldsymbol{w},\lambda)\begin{pmatrix}\frac{\partial p}{\partial \boldsymbol{w}}\\ \frac{\partial p}{\partial\lambda}\end{pmatrix}. \tag{8.50}$$

Based on different information criteria, both Moller and Hou tried to find the inverse of their Hessian $\boldsymbol{H}^{-1}(\boldsymbol{w},\lambda)$. Although the inverse Hessian of Moller and Hou is different, they finally obtained the same CPCA and CMCA rules [5]. Here we propose to derive the differential equation with another technical, which is

$$\boldsymbol{H}(\boldsymbol{w},\lambda)\begin{pmatrix}\dot{\boldsymbol{w}}\\ \dot{\lambda}\end{pmatrix}=-\begin{pmatrix}\frac{\partial p}{\partial \boldsymbol{w}}\\ \frac{\partial p}{\partial\lambda}\end{pmatrix}. \tag{8.51}$$

In this case, there is no need to calculate the inverse Hessian. Substituting (8.44), (8.45), and (8.49) into (8.51), it yields

$$2\begin{pmatrix}\boldsymbol{C}-\lambda\boldsymbol{I} & -\boldsymbol{w}\\ -\boldsymbol{w}^{\mathrm{T}} & 0\end{pmatrix}\begin{pmatrix}\dot{\boldsymbol{w}}\\ \dot{\lambda}\end{pmatrix}=-\begin{pmatrix}2\boldsymbol{C}\boldsymbol{w}-2\lambda\boldsymbol{w}\\ -\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}+1\end{pmatrix}. \tag{8.52}$$

Then we can get

$$(\boldsymbol{C}-\lambda\boldsymbol{I})\dot{\boldsymbol{w}}-\boldsymbol{w}\dot{\lambda}=-(\boldsymbol{C}-\lambda\boldsymbol{I})\boldsymbol{w} \tag{8.53}$$

$$-2\boldsymbol{w}^{\mathrm{T}}\dot{\boldsymbol{w}}=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}-1. \tag{8.54}$$

In the vicinity of the stationary point $(\boldsymbol{w}_1,\lambda_1)$, by approximating $\boldsymbol{w}\approx\boldsymbol{w}_1$, $\lambda\approx\lambda_1\ll\lambda_j$ $(2\le j\le n)$, and after some manipulations (see Appendix A in [6]), we get a coupled dynamical system as

$$\dot{\boldsymbol{w}}=\frac{\boldsymbol{C}^{-1}\boldsymbol{w}(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}+1)}{2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}^{-1}\boldsymbol{w}}-\boldsymbol{w} \tag{8.55}$$

$$\dot{\lambda}=\frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}+1}{2}\left(\frac{1}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}^{-1}\boldsymbol{w}}-\lambda\right). \tag{8.56}$$

8.3.2 The Unified and Coupled Learning Algorithms

8.3.2.1 Coupled MCA Algorithms

The differential equations can be turned into the online form by informally approximating $\boldsymbol{C} = \boldsymbol{x}(k)\boldsymbol{x}^{\mathrm{T}}(k)$, where $\boldsymbol{x}(k)$ is a data vector drawn from the distribution. That is, the expression of the rules in online form can be approximated by slowly following $\boldsymbol{w}(k+1) = f(\boldsymbol{x}(k)\boldsymbol{x}^{\mathrm{T}}(k);\ \boldsymbol{w}(k))$ for subsequent observations of $\boldsymbol{x}$. Moller has pointed out [2] that this transition is infeasible if the equation contains $\boldsymbol{C}^{-1}$, because it is hard to replace the inverse matrix $\boldsymbol{C}^{-1}$ by an expression containing the input vector $\boldsymbol{x}$. However, this problem can be solved in another way [11, 12], in which $\boldsymbol{C}^{-1}$ is updated as

$$\widehat{\boldsymbol{C}}^{-1}(k+1) = \frac{k+1}{k}\left[\widehat{\boldsymbol{C}}^{-1}(k) - \frac{\widehat{\boldsymbol{C}}^{-1}(k)\,\boldsymbol{x}(k+1)\boldsymbol{x}^{\mathrm{T}}(k+1)\,\widehat{\boldsymbol{C}}^{-1}(k)}{k+\boldsymbol{x}^{\mathrm{T}}(k+1)\,\widehat{\boldsymbol{C}}^{-1}(k)\,\boldsymbol{x}^{\mathrm{T}}(k+1)}\right] \tag{8.57}$$

where $\widehat{\boldsymbol{C}}^{-1}(k)$ starts with $\widehat{\boldsymbol{C}}^{-1}(0) = \boldsymbol{I}$ and converges to $\boldsymbol{C}^{-1}$ as $k \to \infty$.Then, the CMCA system (8.55)–(8.56) has the online form as:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \gamma(k)\left\{\frac{[\boldsymbol{w}^{\mathrm{T}}(k)\boldsymbol{w}(k)+1]\,\boldsymbol{Q}(k)\boldsymbol{w}(k)}{2\boldsymbol{w}^{\mathrm{T}}(k)\,\boldsymbol{Q}(k)\boldsymbol{w}(k)} - \boldsymbol{w}(k)\right\} \tag{8.58}$$

$$\lambda(k+1) = \lambda(k) + \gamma(k)\frac{\boldsymbol{w}^{\mathrm{T}}(k)\boldsymbol{w}(k)+1}{2}\left[\frac{1}{\boldsymbol{w}^{\mathrm{T}}(k)\boldsymbol{Q}(k)\boldsymbol{w}(k)} - \lambda(k)\right] \tag{8.59}$$

$$\boldsymbol{Q}(k+1) = \frac{k+1}{\alpha k}\left[\boldsymbol{Q}(k) - \frac{\boldsymbol{Q}(k)\,\boldsymbol{x}(k+1)\,\boldsymbol{x}^{\mathrm{T}}(k+1)\,\boldsymbol{Q}(k)}{k+\boldsymbol{x}^{\mathrm{T}}(k+1)\,\boldsymbol{Q}(k)\,\boldsymbol{x}^{\mathrm{T}}(k+1)}\right] \tag{8.60}$$

where $0<\alpha\leq 1$ denotes the forgetting factor and $\gamma(k)$ is the learning rate. If all training samples come from a stationary process, we choose $\alpha = 1$. $\boldsymbol{Q}(k) = \boldsymbol{C}^{-1}(k)$ starts with $\boldsymbol{Q}(0) = \boldsymbol{I}$. Here, we refer to the rule (8.55)–(8.56) and its online form (8.58)–(8.60) as "fMCA," where f means fast. In the rest of this section, the online form (which is used in the implementation) and the differential matrix form (which is used in the convergence analysis) of a rule have the same name, and we will not emphasize this again. If we further approximate $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w} \approx 1$ (which fulfills in the vicinity of the stationary points) in (8.55)–(8.56), we can obtain q simplified CMCA system

$$\dot{\boldsymbol{w}} = \frac{\boldsymbol{C}^{-1}\boldsymbol{w}}{2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}^{-1}\boldsymbol{w}} - \boldsymbol{w} \tag{8.61}$$

$$\dot{\lambda} = \frac{1}{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}^{-1}\boldsymbol{w}} - \lambda \tag{8.62}$$

and the online form is given as:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \gamma(k)\left\{\frac{\boldsymbol{Q}(k)\,\boldsymbol{w}(k)}{\boldsymbol{w}^{\mathrm{T}}(k)\,\boldsymbol{Q}(k)\,\boldsymbol{w}(k)} - \boldsymbol{w}(k)\right\} \tag{8.63}$$

$$\lambda(k+1) = \lambda(k) + \gamma(k)\left[\frac{1}{\boldsymbol{w}^{\mathrm{T}}(k)\,\boldsymbol{Q}(k)\,\boldsymbol{w}(k)} - \lambda(k)\right] \tag{8.64}$$

where $\mathbf{Q}(k)$ is updated by (8.60). In the following, we will refer to this algorithm as "aMCA," where a means adaptive.

8.3.2.2 Coupled PCA Algorithms

It is known that in unified rules, MCA rules can be derived from PCA rules by changing the sign or using the inverse of the covariance matrix, and vice versa. Here we propose to derive unified algorithms by deriving CPCA rules from CMCA rules. Suppose that the covariance matrix $\mathbf{C}$ has an eigen pair $(\boldsymbol{w}, \lambda)$; then it holds that [13] $\boldsymbol{C}\boldsymbol{w} = \lambda\boldsymbol{w}$ and $\boldsymbol{C}^{-1}\boldsymbol{w} = \lambda^{-1}\boldsymbol{w}$, which means that the minor eigen pair of $\boldsymbol{C}$ is also the principal eigen pair of the inverse matrix $\boldsymbol{C}^{-1}$, and vice versa. Therefore, by replacing $\boldsymbol{C}^{-1}$ with $\boldsymbol{C}$ in fMCA and aMCA rules, respectively, we obtain two modified rules to extract the principal eigen pair of $\boldsymbol{C}$, which is also the minor eigen pair of $\boldsymbol{C}^{-1}$. The modified rules are given as:

$$\dot{\boldsymbol{w}} = \frac{\boldsymbol{C}\boldsymbol{w}\,(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}+1)}{2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w}} - \boldsymbol{w} \tag{8.65}$$

$$\dot{\lambda} = \frac{\boldsymbol{w}^{\mathrm{T}}\boldsymbol{w}+1}{2}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w} - \lambda\right) \tag{8.66}$$

and

$$\dot{\boldsymbol{w}} = \frac{\boldsymbol{C}\boldsymbol{w}}{2\boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w}} - \boldsymbol{w} \tag{8.67}$$

$$\dot{\lambda} = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{C}\boldsymbol{w} - \lambda. \tag{8.68}$$

Since the covariance matrix $\boldsymbol{C}$ is usually unknown in advance, we use its estimate at time k by $\widehat{\boldsymbol{C}}(k)$ suggested in [11], which is

$$\widehat{\boldsymbol{C}}(k+1) = \alpha\frac{k}{k+1}\widehat{\boldsymbol{C}}(k) + \frac{1}{k+1}\boldsymbol{x}(k+1)\boldsymbol{x}^{\mathrm{T}}(k+1) \tag{8.69}$$

where $\widehat{\boldsymbol{C}}(k)$ starts with $\widehat{\boldsymbol{C}}(0) = \boldsymbol{x}(0)\boldsymbol{x}^{\mathrm{T}}(0)$ (or $\boldsymbol{I}$). Actually, (8.57) is obtained from (8.69) by using the SM-formula. Then, the online form of (8.65)–(8.66) and (8.67)–(8.68) is given as:

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \gamma(k)\left\{\frac{\left[\boldsymbol{w}^{\mathrm{T}}(k)\,\widehat{\boldsymbol{C}}(k)\boldsymbol{w}(k) + 1\right]\widehat{\boldsymbol{C}}(k)\boldsymbol{w}(k)}{2\boldsymbol{w}^{\mathrm{T}}(k)\widehat{\boldsymbol{C}}(k)\boldsymbol{w}(k)} - \boldsymbol{w}(k)\right\} \tag{8.70}$$

$$\lambda(k+1) = \lambda(k) + \gamma(k)\frac{\boldsymbol{w}^{\mathrm{T}}(k)\boldsymbol{w}(k) + 1}{2}\left[\boldsymbol{w}^{\mathrm{T}}(k)\widehat{\boldsymbol{C}}(k)\boldsymbol{w}(k) - \lambda(k)\right] \tag{8.71}$$

and

$$\boldsymbol{w}(k+1) = \boldsymbol{w}(k) + \gamma(k)\left\{\frac{\widehat{\boldsymbol{C}}(k)\,\boldsymbol{w}(k)}{\boldsymbol{w}^{\mathrm{T}}(k)\,\widehat{\boldsymbol{C}}(k)\,\boldsymbol{w}(k)} - \boldsymbol{w}(k)\right\} \tag{8.72}$$

$$\lambda(k+1) = \lambda(k) + \gamma(k)\left[\boldsymbol{w}^{\mathrm{T}}(k)\,\widehat{\boldsymbol{C}}(k)\,\boldsymbol{w}(k) - \lambda(k)\right] \tag{8.73}$$

respectively. Here we rename this algorithm deduced from fMCA and aMCA as "fPCA" and "aPCA," respectively. Finally, we obtain two unified and coupled algorithms. The first one is fMCA + fGPCA, and the second one is aMCA + aPCA. These two unified algorithms are capable of both PCA and MCA by using the original or inverse of covariance matrix.

8.3.2.3 Multiple Eigen Pairs Estimation

In some engineering practice, it is required to estimate the eigen-subspace or multiple eigen pairs. As introduced in [4], by using the nested orthogonal complement structure of the eigen-subspace, the problem of estimating the $p(\leq n)$-dimensional principal/minor subspace can be reduced to multiple principal/minor eigenvectors estimation. The following shows how to estimate there maining $p - 1$ principal/minor eigen pairs.

For the CMCA case, consider the following equations:

$$\widehat{\boldsymbol{C}}_j = \widehat{\boldsymbol{C}}_{j-1} + \eta\lambda_{j-1}\boldsymbol{w}_{j-1}\boldsymbol{w}_{j-1}^{\mathrm{T}},\ j = 2, \ldots, p \tag{8.74}$$

where $\widehat{\boldsymbol{C}}_1 = \widehat{\boldsymbol{C}}$ and η is larger than the largest eigenvalue of $\widehat{\boldsymbol{C}}$, and $(\boldsymbol{w}_{j-1}, \lambda_{j-1})$ is the $(j-1)$th minor eigen pair of $\widehat{\boldsymbol{C}}$ that has been extracted. It is found that

$$\begin{aligned}\widehat{\boldsymbol{C}}_j\boldsymbol{w}_q &= (\widehat{\boldsymbol{C}}_{j-1}+\eta\lambda_{j-1}\boldsymbol{w}_{j-1}^{\mathrm{T}})\boldsymbol{w}_q \\ &= (\widehat{\boldsymbol{C}}_1+\eta\sum_{r=1}^{j-1}\lambda_r\boldsymbol{w}_r\boldsymbol{w}_r^{\mathrm{T}})\boldsymbol{w}_q \\ &= \widehat{\boldsymbol{C}}_1\boldsymbol{w}_q+\eta\sum_{r=1}^{j-1}\lambda_r\boldsymbol{w}_r\boldsymbol{w}_r^{\mathrm{T}}\boldsymbol{w}_q \\ &= \begin{cases}\widehat{\boldsymbol{C}}_1\boldsymbol{w}_q+\eta\lambda_q\boldsymbol{w}_q=(1+\eta)\lambda_q\boldsymbol{w}_q & \text{for } q=1,\ldots,j-1\\ \widehat{\boldsymbol{C}}_1\boldsymbol{w}_q=\lambda_q\boldsymbol{w}_q & \text{for } q=j,\ldots,p\end{cases}.\end{aligned} \tag{8.75}$$

Suppose that matrix $\widehat{\boldsymbol{C}}_1$ has eigenvectors $\boldsymbol{w}_1,\boldsymbol{w}_2,\ldots,\boldsymbol{w}_n$ corresponding to eigenvalues $(0<)\ \sigma_1<\sigma_2<\cdots<\sigma_n$, and then matrix $\boldsymbol{C}_j$ has eigenvectors $\boldsymbol{w}_j,\ldots,\boldsymbol{w}_n,\ \boldsymbol{w}_1,\ldots,\boldsymbol{w}_{j-1}$ corresponding to eigenvalues $(0<)\sigma_j<\cdots<\sigma_n$ $<(1+\eta)\sigma_1<\cdots<(1+\eta)\sigma_{j-1}$. In this case, σ_j is the smallest eigenvalue of $\boldsymbol{C}_j$. Based on the SM-formula, we have

$$\begin{aligned}\boldsymbol{Q}_j=\boldsymbol{C}_j^{-1} &= (\boldsymbol{C}_{j-1}+\eta\lambda_{j-1}\boldsymbol{w}_{j-1}\boldsymbol{w}_{j-1}^{\mathrm{T}})^{-1} \\ &= \boldsymbol{C}_{j-1}^{-1}-\frac{\eta\lambda_{j-1}\boldsymbol{C}_{j-1}^{-1}\boldsymbol{w}_{j-1}\boldsymbol{w}_{j-1}^{\mathrm{T}}\boldsymbol{C}_{j-1}^{-1}}{1+\eta\lambda_{j-1}\boldsymbol{w}_{j-1}^{\mathrm{T}}\widehat{\boldsymbol{C}}_{j-1}^{-1}\boldsymbol{w}_{j-1}} \\ &= \boldsymbol{Q}_{j-1}-\frac{\eta\lambda_{j-1}\boldsymbol{Q}_{j-1}\boldsymbol{w}_{j-1}\boldsymbol{w}_{j-1}^{\mathrm{T}}\boldsymbol{Q}_{j-1}}{1+\eta\lambda_{j-1}\boldsymbol{w}_{j-1}^{\mathrm{T}}\boldsymbol{Q}_{j-1}\boldsymbol{w}_{j-1}},\ j=2,\ldots,p.\end{aligned} \tag{8.76}$$

Thus, by replacing $\widehat{\boldsymbol{Q}}$ with $\widehat{\boldsymbol{Q}}_j$ in (8.58)–(8.59) or (8.63)–(8.64), they can be used to estimate the jth minor eigen pair $(\boldsymbol{w}_j,\lambda_j)$ of $\widehat{\boldsymbol{C}}$.

For the CPCA case, consider the following equations

$$\boldsymbol{C}_j=\boldsymbol{C}_{j-1}-\lambda_{j-1}\boldsymbol{w}_{j-1}\boldsymbol{w}_{j-1}^{\mathrm{T}},\ j=2,\ldots,p \tag{8.77}$$

where $(\boldsymbol{w}_{j-1},\lambda_{j-1})$ is the $(j-1)$th principal eigen pair that has been extracted. It is found that

$$\begin{aligned}\widehat{\boldsymbol{C}}_j\boldsymbol{w}_q &= (\widehat{\boldsymbol{C}}_{j-1}-\lambda_{j-1}\boldsymbol{w}_{j-1}\boldsymbol{w}_{j-1}^{\mathrm{T}})\boldsymbol{w}_q \\ &= (\widehat{\boldsymbol{C}}_1-\sum_{r=1}^{j-1}\lambda_r\boldsymbol{w}_r\boldsymbol{w}_r^{\mathrm{T}})\boldsymbol{w}_q \\ &= \widehat{\boldsymbol{C}}_1\boldsymbol{w}_q-\sum_{r=1}^{j-1}\lambda_r\boldsymbol{w}_r\boldsymbol{w}_r^{\mathrm{T}}\boldsymbol{w}_q \\ &= \begin{cases}0 & \text{for } q=1,\ldots,j-1\\ \widehat{\boldsymbol{C}}_1\boldsymbol{w}_q=\lambda_q\boldsymbol{w}_q & \text{for } q=j,\ldots,p\end{cases}.\end{aligned} \tag{8.78}$$

Suppose that the matrix $\widehat{\boldsymbol{C}}_1$ has eigenvectors $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n$ corresponding to eigenvalues $\sigma_1 > \sigma_2 > \cdots > \sigma_n (> 0)$, and then the matrix $\boldsymbol{C}_j$ has eigenvectors $\boldsymbol{w}_j, \ldots, \boldsymbol{w}_n, \boldsymbol{w}_1, \ldots, \boldsymbol{w}_{j-1}$ corresponding to eigenvalues $\sigma_j > \cdots > \sigma_n > \hat{\sigma}_1 = \cdots = \hat{\sigma}_{j-1} (= 0)$. In this case, σ_j is the largest eigenvalue of $\boldsymbol{C}_j$. Thus, by replacing $\widehat{\boldsymbol{C}}$ with $\widehat{\boldsymbol{C}}_j$ in (8.70)–(8.71) or (8.72)–(8.73), they can be used to estimate the jth principal eigen pair $(\boldsymbol{w}_j, \lambda_j)$ of $\widehat{\boldsymbol{C}}$.

8.3.3 Analysis of Convergence and Self-stabilizing Property

The major work of convergence analysis of coupled rules is to find the eigenvalues of the Jacobian

$$\boldsymbol{J}(\boldsymbol{w}_1, \lambda_1) = \begin{pmatrix} \frac{\partial \dot{\boldsymbol{w}}}{\partial \boldsymbol{w}^{\mathrm{T}}} & \frac{\partial \dot{\boldsymbol{w}}}{\partial \lambda} \\ \frac{\partial \dot{\lambda}}{\partial \boldsymbol{w}^{\mathrm{T}}} & \frac{\partial \dot{\lambda}}{\partial \lambda} \end{pmatrix} \tag{8.79}$$

of the differential equations for a stationary point $(\boldsymbol{w}_1, \lambda_1)$. For fMCA rule, after some manipulations (see Appendix B in [6]), we get

$$\boldsymbol{J}_{fMCA}(\boldsymbol{w}_1, \lambda_1) = \begin{pmatrix} \boldsymbol{C}^{-1}\lambda_1 - \boldsymbol{I} - \boldsymbol{w}_1\boldsymbol{w}_1^{\mathrm{T}} & \boldsymbol{0} \\ -2\lambda_1 \boldsymbol{w}_1^{\mathrm{T}} & -1 \end{pmatrix}. \tag{8.80}$$

The Jacobian can be simplified by an orthogonal transformation with

$$\boldsymbol{U} = \begin{pmatrix} \overline{\boldsymbol{W}} & \boldsymbol{0} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{pmatrix}. \tag{8.81}$$

The transformed Jacobian $\boldsymbol{J}^* = \boldsymbol{U}^{\mathrm{T}}\boldsymbol{J}\boldsymbol{U}$ has the same eigenvalues as $\boldsymbol{J}$. In the vicinity of a stationary point $(\boldsymbol{w}_1, \lambda_1)$, we approximate $\overline{\boldsymbol{W}}^{\mathrm{T}}\boldsymbol{w} \approx \boldsymbol{e}_1$ and obtain

$$\boldsymbol{J}^*_{fMCA}(\boldsymbol{w}_1, \lambda_1) = \begin{pmatrix} \bar{\boldsymbol{\Lambda}}^{-1}\lambda_1 - \boldsymbol{I} - \boldsymbol{e}_1\boldsymbol{e}_1^{\mathrm{T}} & \boldsymbol{0} \\ -2\lambda_1 \boldsymbol{e}_1^{\mathrm{T}} & -1 \end{pmatrix}. \tag{8.82}$$

The eigenvalues α of $\boldsymbol{J}^*$ are determined as $\det(\boldsymbol{J}^* - \alpha\boldsymbol{I}) = 0$, which are

$$\alpha_1 = \alpha_{n+1} = -1,\ \alpha_j = \frac{\lambda_1}{\lambda_j} - 1 \overset{\lambda_1 \ll \lambda_j}{\approx} -1,\ j = 2, \ldots, n. \tag{8.83}$$

Since stability requires $\alpha < 0$ and thus $\lambda_1 < \lambda_j$, $j = 2, \ldots, n$, we find that only minor eigen pairs are stable stationary points, while all others are saddles or repellers. What's more, if we further assume $\lambda_1 \ll \lambda_j$, all eigenvalues are $\alpha \approx -1$. Hence, the system converges with approximately equal speed in all its eigen

directions, and this speed is widely independent of the eigenvalues λ_j of the covariance matrix [2]. That is to say, the speed stability problem does not exist in fMCA algorithm.

Similarly, for aMCA rule, we analyze the stability by finding the eigenvalues of

$$\boldsymbol{J}^*_{aMCA}(\boldsymbol{w}_1, \lambda_1) = \begin{pmatrix} \bar{\boldsymbol{\Lambda}}^{-1}\lambda_1 - \boldsymbol{I} - 2\boldsymbol{e}_1\boldsymbol{e}_1^{\mathrm{T}} & \boldsymbol{0} \\ -2\lambda_1\boldsymbol{e}_1^{\mathrm{T}} & -1 \end{pmatrix} \tag{8.84}$$

which are

$$\alpha_1 = -2,\ \alpha_{n+1} = -1,\ \alpha_j = \frac{\lambda_1}{\lambda_j} - 1,\ j = 2, \ldots, n. \tag{8.85}$$

The situation of aMCA is similar to that of fMCA, and the only difference is that the first eigenvalue of Jacobian is $\alpha_1 = -1$ for fMCA and $\alpha_1 = -2$ for aMCA. Thus, the convergence speed of fMCA and aMCA is almost the same.

Similarly, the transformed Jacobian functions of fPCA and aPCA are given as:

$$\boldsymbol{J}^*_{fPCA}(\boldsymbol{w}_1, \lambda_1) = \begin{pmatrix} \bar{\boldsymbol{\Lambda}}^{-1}\lambda_1 - \boldsymbol{I} - \boldsymbol{e}_1\boldsymbol{e}_1^{\mathrm{T}} & \boldsymbol{0} \\ 2\lambda_1\boldsymbol{e}_1^{\mathrm{T}} & -1 \end{pmatrix} \tag{8.86}$$

and

$$\boldsymbol{J}^*_{aPCA}(\boldsymbol{w}_1, \lambda_1) = \begin{pmatrix} \bar{\boldsymbol{\Lambda}}^{-1}\lambda_1 - \boldsymbol{I} - 2\boldsymbol{e}_1\boldsymbol{e}_1^{\mathrm{T}} & \boldsymbol{0} \\ 2\lambda_1\boldsymbol{e}_1^{\mathrm{T}} & -1 \end{pmatrix} \tag{8.87}$$

respectively. And the eigenvalues of (8.86) and (8.87) are given as:

$$\alpha_1 = \alpha_{n+1} = -1,\ \alpha_j = \frac{\lambda_j}{\lambda_n} - 1 \overset{\lambda_n \gg \lambda_j}{\approx} -1,\ j = 1, \ldots, n-1 \tag{8.88}$$

$$\alpha_1 = -2, \alpha_{n+1} = -1,\ \alpha_j = \frac{\lambda_j}{\lambda_n} - 1 \overset{\lambda_n \ll \lambda_j}{\approx} -1,\ j = 1, \ldots, n-1 \tag{8.89}$$

respectively. We can see that only principal eigen pairs are stable stationary points, while all others are saddles or repellers. We can further assume $\lambda_1 \gg \lambda_j$ and thus $\alpha_j \approx -1$ $(j \neq 1)$ for fPCA and aPCA.

The analysis of the self-stabilizing property of the proposed algorithms is omitted here. For details, see [6].

8.3.4 Simulation Experiments

In this section, we provide several experiments to illustrate the performance of the proposed algorithms in comparison with some well-known coupled algorithms and unified algorithms. Experiments 1 and 2 mainly show the stability of proposed CMCA and CPCA algorithms in comparison with existing CMCA and CPCA algorithms, respectively. In experiment 3, the self-stabilizing property of the proposed algorithm is shown. In experiment 4, we compare the performance of aMCA and aPCA with that of two unified algorithms. Experiments 5 and 6 illustrate some examples of practical applications.

In experiments 1–4, all algorithms are used to extract the minor or principal component from a high-dimensional input data sequence, which is generated from $\boldsymbol{x} = \boldsymbol{B} \cdot \boldsymbol{y}(t)$, where each column of $\boldsymbol{B} \in \Re^{30\times 30}$ is Gaussian with variance 1/30, and $\boldsymbol{y}(t) \in \Re^{30\times 1}$ is Gaussian and randomly generated.

In all experiments, to measure the estimation accuracy, we compute the norm of eigenvector estimation (weight vector) $\|\boldsymbol{w}(k)\|$ and the projection $[\psi(k)]$ of the weight vector onto the true eigenvector at each step:

$$\psi(k) = \frac{|\boldsymbol{w}^{\mathrm{T}}(k)\boldsymbol{w}_1|}{\|\boldsymbol{w}(k)\|}$$

where $\boldsymbol{w}_1$ is the true minor (for MCA) or principal (for PCA) eigenvector with unit length.

Unless otherwise stated, we set the initial conditions of experiments 1–4 as follows: (1) The weight vector is initialized with a random vector (unit length). (2) The learning rate $\gamma(k)$ starts at $\gamma(0) = 10^{-2}$ and decays exponentially toward zero with a final value $\gamma(k_{\max}) = 10^{-4}$. (3) We set $\alpha = 1$ (if used), and $\lambda(0) = 0.001$ for all cMCA and cPCA algorithms.

In experiments 1 and 2, $k_{\max}$ = 20,000 training steps are executed for all algorithms. In order to test the stability of the proposed algorithms, after 10,000 training steps, we drastically change the input signals; thus, the eigen information changed suddenly. All algorithms start to extract the new eigen pair since k = 10001. The learning rate for nMCA is 10 times smaller than that for the others. Then, 20 times of Monte Carlo simulation are executed for all experiments.

Figure 8.1 shows the time course of the projection of minor weight vector. We can see that in all rules except mMCA the projection converges toward unity; thus, these weight vectors align with the true eigenvector. The convergence speed of mMCA is lower than that of the others and the projection of mMCA cannot converge toward unity within 10,000 steps. We can also find that the convergence speed of fMCA and aMCA rules is similar, and higher than that of the others. We can also find that, at time step k = 10,001, where the input signals changed suddenly, all algorithms start to extract the new eigen pair. Figure 8.2 shows the time course of weight vector length. We can find that the vector length of nMCA converges to a nonunit length. The convergence speed and the stability of fMCA

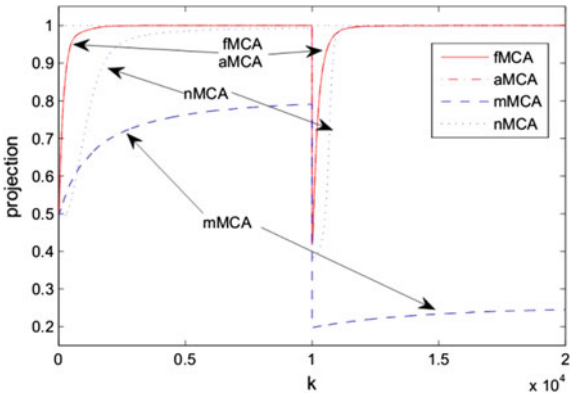

Fig. 8.1 Projection of weight vector onto the true minor eigenvector

and aMCA are higher and better than that of the others. It can be seen that the convergence speed of aMCA is a bit higher than that of fMCA.

Figure 8.3 shows the time course of the minor eigenvalue estimation. We can see that mMCA cannot extract the minor eigenvalue as effective as the other algorithms after the input signals changed. From Figs. 8.1 to 8.3, we can conclude that the performance of fMCA and aMCA is better than that of the other cMCA algorithms. Moreover, nMCA contains $\boldsymbol{C}$ and $\boldsymbol{C}^{-1}$ simultaneously in the equations, and we can prove that mMCA also has the speed stability problem though it is a coupled rule. These may be the reason why our algorithms perform better than nMCA and mCMA.

In experiment 2, we compare the performance of fPCA and aPCA with that of ALA and nPCA. The time course of the projection and the eigenvector length of principal weight vector are shown in Figs. 8.4 and 8.5, and the principal eigenvalue estimation is shown in Fig. 8.6, respectively. In Fig. 8.5, the curves for fPCA and aPCA are shown in a subfigure because of its small amplitude. We can see that the convergence speed of fPCA and aPCA is similar to that of nPCA and ALA, but fPCA and aPCA have less fluctuations over time compared with nPCA and ALA.

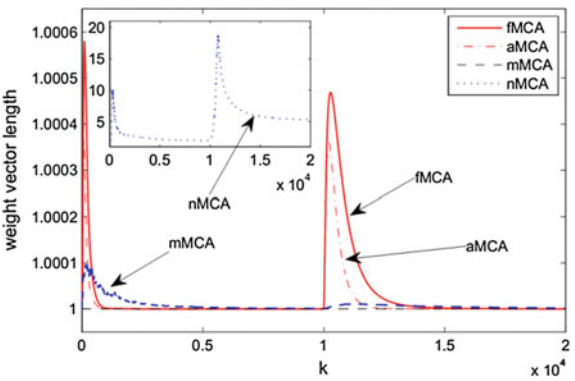

Fig. 8.2 Weight vector length

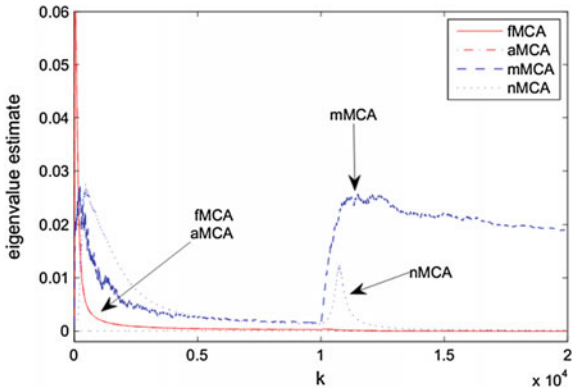

Fig. 8.3 Minor eigenvalue estimation

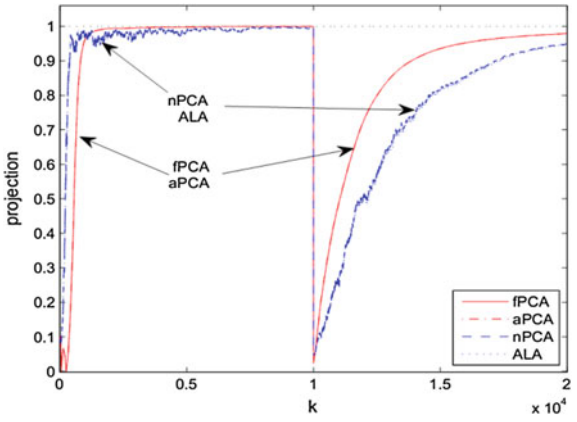

Fig. 8.4 Projection of weight vector onto the true principal eigenvector

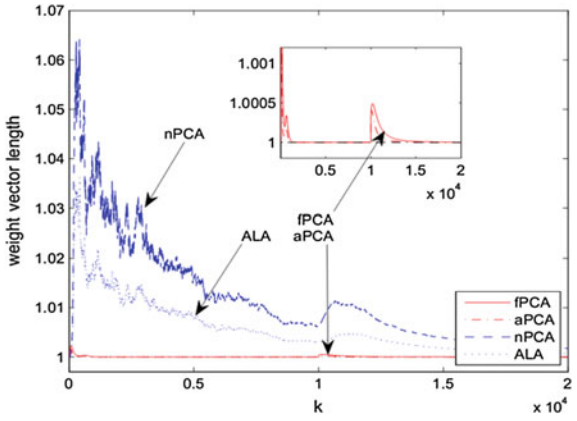

Fig. 8.5 Weight vector length

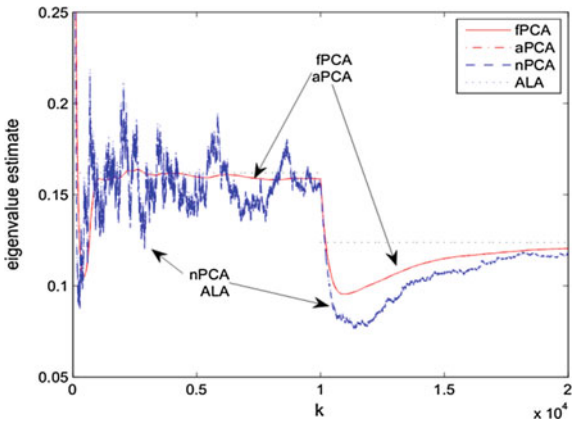

Fig. 8.6 Principal eigenvalue estimation

This is actually because that in fPCA and aPCA the covariance matrix $\boldsymbol{C}$ is updated by (8.69) while in nPCA and ALA$\mathbf{C}$ that is updated by $\boldsymbol{C}(k) = \boldsymbol{x}(k)\,\boldsymbol{x}^{\mathrm{T}}(k)$.

Experiment 3 is used to test the self-stabilizing property of the proposed algorithms. Figure 8.7 shows the time course of weight vector length estimation of fMCA, aMCA, fPCA, and aPCA which are initialized with nonunit length. We can find that all algorithms converge to unit length rapidly, which shows the self-stabilizing property of eigenvector estimates. The self-stabilizing property of eigenvalue estimates is shown in Figs. 8.3 and 8.6. From the results of experiments 1–3, we can see that the performance off MCA and fPCA is similar to that of aMCA and aPCA, respectively. Thus in experiment 4, we only compare the performance of aMCA and aPCA with that of two unified algorithms which were proposed in recent years, i.e., (1) kMCA + kPCA [14], where k means this algorithm was

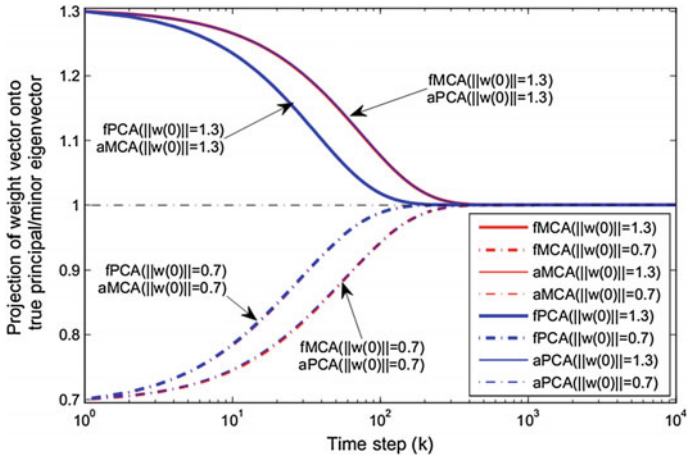

Fig. 8.7 Weight vector length

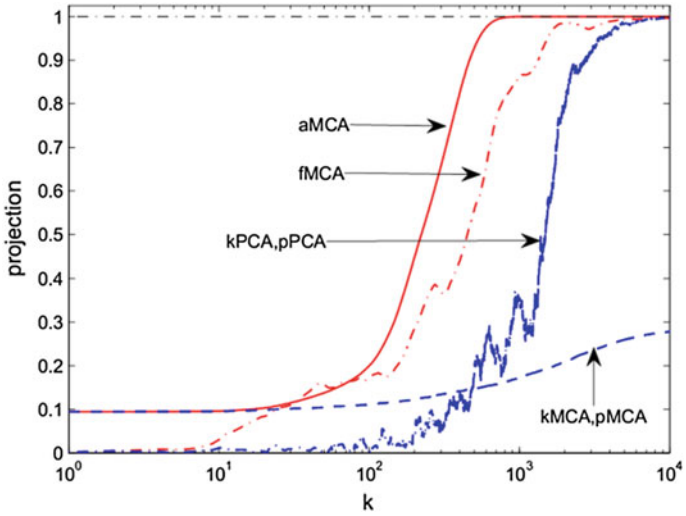

Fig. 8.8 Projection of weight vector onto the true principal/minor eigenvector

proposed by Kong;(2) *p*MCA + *p*PCA [15], where *p* means this algorithm was proposed by Peng. The time course of the projection of weight vector onto the true principal/minor eigenvector and the weight vector length is shown in Figs. 8.8 and 8.9, respectively. In Fig. 8.9, the first 1000 steps of aMCA and kMCA are shown in a subfigure. We can see that the proposed algorithms perform better the existing unified algorithms.

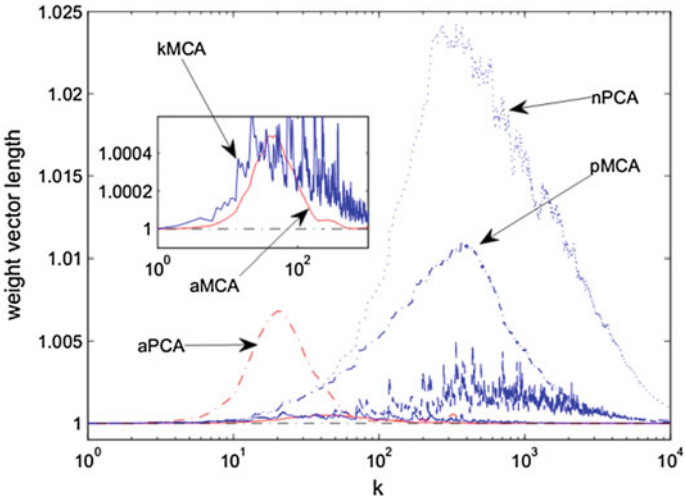

Fig. 8.9 Weight vector length

In summary, we propose a novel method to derive neural network algorithms based on a special information criterion. We firstly obtain two CMCA algorithms based on the modified Newton's method. Then, two CPCA rules are obtained from the CMCA rules. In this case, two unified and coupled algorithms are obtained, which are capable of both PCA and MCA and can also mitigate the speed-stability problem. The proposed algorithms converge faster and are more stable than existing algorithms. Moreover, all of the proposed algorithms are self-stabilized.

8.4 Adaptive Coupled Generalized Eigen Pairs Extraction Algorithms

In [4], based on Moller's work, Nguyen developed two well-performed quasi-Newton-type algorithms to extract generalized eigen pairs. Actually, Nguyen's algorithms are the generalization of Moller's coupled learning algorithms. But with DDT approach, Nguyen also reported the explicit convergence analysis for their learning rules, i.e., the region within which the initial estimate of the eigen pair must be chosen to guarantee the convergence to the desired eigen pair. However, as stated in [4], the GMCA algorithm proposed in [4] may lose robustness when the smallest eigenvalue of the matrix pencil is far less than 1.

Motivated by the efficacy of the coupled learning rules in [2] and [4] for the HEP and GHEP, we will introduce novel coupled algorithms proposed by us to estimate the generalized eigen pair information in this section. Based on a novel generalized information criterion, we have obtained an adaptive GMCA algorithm, as well as an adaptive GPCA algorithm by modifying the GMCA algorithm. It is worth noting that the procedure of obtaining the algorithms in this section is easier than the existing methods, for that it does not need to calculate the inverse of the Hessian matrix when deriving the new algorithms. It can be seen that our algorithms do not involve the reciprocal of the estimated eigenvalue in equations. Thus, they are numerically more robust than Nguyen's algorithms even when the smallest eigenvalue of the matrix pencil is far less than 1. Compared with Nguyen's algorithms, it is much easier to choose step size for online implementation of the algorithms.

8.4.1 A Coupled Generalized System for GMCA and GPCA

A. Generalized information criterion and coupled generalized system

Generally speaking, neural network model-based algorithms are often derived by optimizing some cost function or information criterion [2, 16]. As pointed out in [17], any criterion may be used if the maximum or minimum (possibly under a constraint) coincides with the desired principal or minor directions or subspace. In [2], Moller pointed out that the freedom of choosing an information criterion is

greater if Newton's method is applied. In that case, it suffices to find a criterion of which the stationary points coincide with the desired solutions. Moller first proposed a special criterion which involves both eigenvector and eigenvalue estimates [2]. Based on Moller's work, Nguyen [4] first proposed to derive novel generalized eigen pair extraction algorithms by finding the stationary points of a generalized information criterion which is actually the generalization of Moller's information criterion.

In this section, for a given matrix pencil ($\boldsymbol{R}_y$, $\boldsymbol{R}_x$), we propose a generalized information criterion based on the criteria introduced in [2] and [4] as

$$p(\boldsymbol{w}, \lambda) = \boldsymbol{w}^H \boldsymbol{R}_y \boldsymbol{w} - \lambda \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} + \lambda. \tag{8.90}$$

We can see that

$$\begin{pmatrix} \frac{\partial p}{\partial w} \\ \frac{\partial p}{\partial \lambda} \end{pmatrix} = \begin{pmatrix} 2\boldsymbol{R}_y \boldsymbol{w} - 2\lambda \boldsymbol{R}_x \boldsymbol{w} \\ -\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} + 1 \end{pmatrix}. \tag{8.91}$$

Thus, the stationary points $(\bar{\boldsymbol{w}}, \bar{\lambda})$ are defined by

$$\begin{cases} \boldsymbol{R}_y \bar{\boldsymbol{w}} = \bar{\lambda} \boldsymbol{R}_x \bar{\boldsymbol{w}} \\ \bar{\boldsymbol{w}}^H \boldsymbol{R}_x \bar{\boldsymbol{w}} = 1 \end{cases}, \tag{8.92}$$

from which we can conclude that $\bar{\boldsymbol{w}}^H \boldsymbol{R}_y \bar{\boldsymbol{w}} = \bar{\lambda} \bar{\boldsymbol{w}}^H \boldsymbol{R}_x \bar{\boldsymbol{w}} = \bar{\lambda}$. These imply that a stationary point $(\bar{\boldsymbol{w}}, \bar{\lambda})$ of (8.90) is a generalized eigen pair of the matrix pencil ($\boldsymbol{R}_y$, $\boldsymbol{R}_x$). The Hessian of the criterion is given as:

$$\boldsymbol{H}(\boldsymbol{w}, \lambda) = \begin{pmatrix} \frac{\partial^2 p}{\partial \boldsymbol{w}^2} & \frac{\partial^2 p}{\partial \boldsymbol{w} \partial \lambda} \\ \frac{\partial^2 p}{\partial \lambda \partial \boldsymbol{w}} & \frac{\partial^2 p}{\partial \lambda^2} \end{pmatrix} = 2 \begin{pmatrix} \boldsymbol{R}_y - \lambda \boldsymbol{R}_x & -\boldsymbol{R}_x \boldsymbol{w} \\ -\boldsymbol{w}^H \boldsymbol{R}_x & 0 \end{pmatrix}. \tag{8.93}$$

After applying the Newton's method, the equation used to obtain the system can be written as:

$$\begin{pmatrix} \dot{\boldsymbol{w}} \\ \dot{\lambda} \end{pmatrix} = -\boldsymbol{H}^{-1}(\boldsymbol{w}, \lambda) \begin{pmatrix} \frac{\partial p}{\partial \boldsymbol{w}} \\ \frac{\partial p}{\partial \lambda} \end{pmatrix}, \tag{8.94}$$

where $\dot{\boldsymbol{w}}$ and $\dot{\lambda}$ are the derivatives of $\boldsymbol{w}$ and λ with respect to time t, respectively. Based on the above equation, Nguyen [4] obtained their algorithms by finding the inverse matrix of the Hessian $\boldsymbol{H}^{-1}(\boldsymbol{w}, \lambda)$. Premultiplying both sides of the above equation by $\boldsymbol{H}(\boldsymbol{w}, \lambda)$, it yields

$$\boldsymbol{H}(\boldsymbol{w}, \lambda) \begin{pmatrix} \dot{\boldsymbol{w}} \\ \dot{\lambda} \end{pmatrix} = -\begin{pmatrix} \frac{\partial p}{\partial \boldsymbol{w}} \\ \frac{\partial p}{\partial \lambda} \end{pmatrix}. \tag{8.95}$$

In this section, all our later algorithms are built on this newly proposed Eq. (8.95). Substituting (8.91) and (8.93) into (8.95), we get

$$2\begin{pmatrix} \boldsymbol{R}_y - \lambda \boldsymbol{R}_x & -\boldsymbol{R}_x \boldsymbol{w} \\ -\boldsymbol{w}^H \boldsymbol{R}_x & 0 \end{pmatrix} \begin{pmatrix} \dot{\boldsymbol{w}} \\ \dot{\lambda} \end{pmatrix} = -\begin{pmatrix} 2\boldsymbol{R}_y \boldsymbol{w} - 2\lambda \boldsymbol{R}_x \boldsymbol{w} \\ -\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} + 1 \end{pmatrix}. \tag{8.96}$$

From (8.96), we can get

$$(\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)\dot{w} - \boldsymbol{R}_x \boldsymbol{w}\dot{\lambda} = -(\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)\boldsymbol{w} \tag{8.97}$$

$$-2\boldsymbol{w}^H \boldsymbol{R}_x \dot{\boldsymbol{w}} = \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} - 1. \tag{8.98}$$

Premultiplying both sides of (8.97) by $(\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1}$ gives the following:

$$\dot{\boldsymbol{w}} = (\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1} \boldsymbol{R}_x \boldsymbol{w}\dot{\lambda} - \boldsymbol{w}. \tag{8.99}$$

Substituting (8.99) into (8.98), we have

$$-2\boldsymbol{w}^H \boldsymbol{R}_x \left((\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1} \boldsymbol{R}_x \boldsymbol{w}\dot{\lambda} - \boldsymbol{w} \right) = \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} - 1. \tag{8.100}$$

Thus,

$$\dot{\lambda} = \frac{\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} + 1}{2\boldsymbol{w}^H \boldsymbol{R}_x (\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1} \boldsymbol{R}_x \boldsymbol{w}}. \tag{8.101}$$

Substituting (8.101) into (8.99), we get

$$\dot{\boldsymbol{w}} = \frac{(\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1} \boldsymbol{R}_x \boldsymbol{w} \left(\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} + 1 \right)}{2\boldsymbol{w}^H \boldsymbol{R}_x \left(\boldsymbol{R}_y - \lambda \boldsymbol{R}_x \right)^{-1} \boldsymbol{R}_x \boldsymbol{w}} - \boldsymbol{w}. \tag{8.102}$$

By approximating $\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{w} = 1$ in the vicinity of the stationary point $(\boldsymbol{w}_1, \lambda_1)$, we get a coupled generalized system as:

$$\dot{\boldsymbol{w}} = \frac{(\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1} \boldsymbol{R}_x \boldsymbol{w}}{\boldsymbol{w}^H \boldsymbol{R}_x (\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1} \boldsymbol{R}_x \boldsymbol{w}} - \boldsymbol{w}, \tag{8.103}$$

$$\dot{\lambda} = \frac{1}{\boldsymbol{w}^H \boldsymbol{R}_x (\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1} \boldsymbol{R}_x \boldsymbol{w}} - \lambda. \tag{8.104}$$

B. Coupled generalized systems for GMCA and GPCA

Let $\boldsymbol{\Lambda}$ be a diagonal matrix containing all generalized eigenvalues of the matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$, i.e., $\boldsymbol{\Lambda} = \text{diag}\{\lambda_1, \ldots, \lambda_N\}$. Let $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N]$, where $\boldsymbol{v}_1$, ..., $\boldsymbol{v}_N$

are the generalized eigenvectors associated with the generalized eigenvalues $\lambda_1, \ldots, \lambda_N$. It holds that $\boldsymbol{V}^H \boldsymbol{R}_x \boldsymbol{V} = \boldsymbol{I}$, $\boldsymbol{V}^H \boldsymbol{R}_y \boldsymbol{V} = \boldsymbol{\Lambda}$. Hence, $\boldsymbol{R}_x = (\boldsymbol{V}^H)^{-1} \boldsymbol{V}^{-1}$ and $\boldsymbol{R}_y = (\boldsymbol{V}^H)^{-1} \boldsymbol{\Lambda} \boldsymbol{V}^{-1}$, and

$$(\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1} = \boldsymbol{V}(\boldsymbol{\Lambda} - \lambda \boldsymbol{I})^{-1} \boldsymbol{V}^H. \tag{8.105}$$

If we consider $\boldsymbol{w} \approx \boldsymbol{v}_1$ and $\lambda \approx \lambda_1 \ll \lambda_j (2 \leq j \leq N)$ in the vicinity of the stationary point $(\boldsymbol{w}_1, \lambda_1)$, then we have $\lambda_j - \lambda \approx \lambda_j$. In that case, $\boldsymbol{V}^H \boldsymbol{R}_x \boldsymbol{w} \approx e_1 = [1, 0, \ldots, 0]^H$ and

$$\begin{aligned} \boldsymbol{\Lambda} - \lambda \boldsymbol{I} &= \operatorname{diag}\{\lambda_1 - \lambda, \ldots, \lambda_N - \lambda\} \\ &\approx \operatorname{diag}\{\lambda_1 - \lambda, \lambda_2, \ldots, \lambda_N\} \\ &= \boldsymbol{\Lambda} - \lambda \boldsymbol{e}_1 \boldsymbol{e}_1^H, \end{aligned} \tag{8.106}$$

where $\operatorname{diag}\{\bullet\}$ is the diagonal function. Substituting (8.106) into (8.105), we get the following:

$$\begin{aligned} (\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1} &= \boldsymbol{V}(\boldsymbol{\Lambda} - \lambda \boldsymbol{I})^{-1} \boldsymbol{V}^H \\ &\approx [(\boldsymbol{V}^H)^{-1} (\boldsymbol{\Lambda} - \lambda \boldsymbol{e}_1 \boldsymbol{e}_1^H) \boldsymbol{V}^{-1}]^{-1} \\ &= [\boldsymbol{R}_y - \lambda (\boldsymbol{V}^H)^{-1} \boldsymbol{e}_1 \boldsymbol{e}_1^H \boldsymbol{V}^{-1}]^{-1} \\ &\approx \left[\boldsymbol{R}_y - \lambda (\boldsymbol{V}^H)^{-1} (\boldsymbol{V}^H \boldsymbol{R}_x \boldsymbol{w})(\boldsymbol{V}^H \boldsymbol{R}_x \boldsymbol{w})^H \boldsymbol{V}^{-1}\right]^{-1} \\ &= \left[\boldsymbol{R}_y - \lambda (\boldsymbol{R}_x \boldsymbol{w})(\boldsymbol{R}_x \boldsymbol{w})^H\right]^{-1}. \end{aligned} \tag{8.107}$$

It can be seen that

$$\begin{aligned} &\left[\boldsymbol{R}_y - \lambda_1 (\boldsymbol{R}_x \boldsymbol{v}_1)(\boldsymbol{R}_x \boldsymbol{v}_1)^H\right] \boldsymbol{v}_1 \\ &\quad = \boldsymbol{R}_y \boldsymbol{v}_1 - (\lambda_1 \boldsymbol{R}_x \boldsymbol{v}_1)(\boldsymbol{v}_1^H \boldsymbol{R}_x \boldsymbol{v}_1) = 0. \end{aligned} \tag{8.108}$$

Since $\boldsymbol{R}_y \boldsymbol{v}_1 = \lambda_1 \boldsymbol{R}_x \boldsymbol{v}_1$ and $\boldsymbol{v}_1^H \boldsymbol{R}_x \boldsymbol{v}_1 = 1$. This means that matrix $\boldsymbol{R}_y - \lambda (\boldsymbol{R}_x \boldsymbol{w})(\boldsymbol{R}_x \boldsymbol{w})^H$ has an eigenvalue 0 associated with eigenvector $\boldsymbol{v}_1$. This is to say, the matrix $\boldsymbol{R}_y - \lambda (\boldsymbol{R}_x \boldsymbol{w})(\boldsymbol{R}_x \boldsymbol{w})^H$ is rank-deficient and hence cannot be inverted if $(\boldsymbol{w}, \lambda) = (\boldsymbol{v}_1, \lambda_1)$. To address this issue, we add a penalty factor $\varepsilon \approx 1$ in (8.107), and then it yields the following:

$$\begin{aligned} (\boldsymbol{R}_y - \lambda \boldsymbol{R}_x)^{-1} &\approx \left[\boldsymbol{R}_y - \varepsilon \lambda (\boldsymbol{R}_x \boldsymbol{w})(\boldsymbol{R}_x \boldsymbol{w})^H\right]^{-1} \\ &= \boldsymbol{R}_y^{-1} + \frac{\varepsilon \lambda \boldsymbol{R}_y^{-1} \boldsymbol{R}_x \boldsymbol{w} \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{R}_y^{-1}}{1 - \varepsilon \lambda \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{R}_y^{-1} \boldsymbol{R}_x \boldsymbol{w}}, \end{aligned} \tag{8.109}$$

The last step of (8.109) is obtained by using the SM-formula (Sherman–Morrison formula) [13]. Substituting (8.107) into (8.103), we get the following:

$$\dot{\boldsymbol{w}} = \frac{\left(\boldsymbol{R}_y^{-1} + \frac{\varepsilon\lambda \boldsymbol{R}_y^{-1}\boldsymbol{R}_x \boldsymbol{w}\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{R}_y^{-1}}{1-\varepsilon\lambda \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{R}_y^{-1}\boldsymbol{R}_x \boldsymbol{w}}\right)\boldsymbol{R}_x \boldsymbol{w}}{\boldsymbol{w}^H \boldsymbol{R}_x \left(\boldsymbol{R}_y^{-1} + \frac{\varepsilon\lambda \boldsymbol{R}_y^{-1}\boldsymbol{R}_x \boldsymbol{w}\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{R}_y^{-1}}{1-\varepsilon\lambda \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{R}_y^{-1}\boldsymbol{R}_x \boldsymbol{w}}\right)\boldsymbol{R}_x \boldsymbol{w}} - \boldsymbol{w}. \tag{8.110}$$

Multiplying the numerator and denominator of (8.110) by $1 - \varepsilon\lambda \boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{R}_y^{-1} \boldsymbol{R}_x \boldsymbol{w}$ simultaneously, and after some manipulations, we get

$$\dot{\boldsymbol{w}} = \frac{\boldsymbol{R}_y^{-1}\boldsymbol{R}_x \boldsymbol{w}}{\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{R}_y^{-1}\boldsymbol{R}_x \boldsymbol{w}} - \boldsymbol{w}. \tag{8.111}$$

Similarly, substituting (8.107) into (8.104), we can get

$$\dot{\lambda} = \frac{1}{\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{R}_y^{-1}\boldsymbol{R}_x \boldsymbol{w}} - \varepsilon\lambda. \tag{8.112}$$

It can be seen that the penalty factor ε is not necessarily needed in the equations. Or in other words, we can approximate $\varepsilon = 1$ in future equations. Thus, we get the following:

$$\dot{\lambda} = \frac{1}{\boldsymbol{w}^H \boldsymbol{R}_x \boldsymbol{R}_y^{-1}\boldsymbol{R}_x \boldsymbol{w}} - \lambda. \tag{8.113}$$

Thus, (8.111) and (8.113) are the coupled systems for the GMCA case.

It is known that the ith principal generalized eigenvector $\boldsymbol{v}_i$ of the matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ is also the ith minor generalized eigenvector of the matrix pencil $(\boldsymbol{R}_x, \boldsymbol{R}_y)$. Hence, the problem of extracting principal generalized subspace of the inversed matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ is equivalent to that of extracting minor generalized subspace of the matrix pencil $(\boldsymbol{R}_x, \boldsymbol{R}_y)$, and vice versa [4]. Therefore, by swapping $\boldsymbol{R}_x$ and $\boldsymbol{R}_y$, $\boldsymbol{R}_x^{-1}$ and $\boldsymbol{R}_y^{-1}$ in (8.111) and (8.113), we obtain a modified system

$$\dot{\boldsymbol{w}} = \frac{\boldsymbol{R}_x^{-1}\boldsymbol{R}_y \boldsymbol{w}}{\boldsymbol{w}^H \boldsymbol{R}_y \boldsymbol{R}_x^{-1}\boldsymbol{R}_y \boldsymbol{w}} - \boldsymbol{w}, \tag{8.114}$$

$$\dot{\lambda} = \boldsymbol{w}^H \boldsymbol{R}_y \boldsymbol{R}_x^{-1}\boldsymbol{R}_y \boldsymbol{w} - \lambda, \tag{8.115}$$

to extract the minor eigen pair of matrix pencil $(\boldsymbol{R}_x, \boldsymbol{R}_y)$ as well as the principal eigen pair of matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$.

As was pointed out in [4], by using the nested orthogonal complement structure of the generalized eigen-subspace, the problem of estimating the $p\,(\leq N)$-dimensional minor/principal generalized subspace can be reduced to multiple GHEPs of

estimating the generalized eigen pairs associated with the smallest/largest generalized eigenvalues of certain matrix pencils. In the following, we will show how to estimate the remaining $p-1$ minor/principal eigen pairs. In the GMCA case, consider the following equations:

$$\boldsymbol{R}_j = \boldsymbol{R}_{j-1} + \rho \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{w}_{j-1} \boldsymbol{w}_{j-1}^{\mathrm{T}} \boldsymbol{R}_{\boldsymbol{y}}, \tag{8.116}$$

$$\boldsymbol{R}_j^{-1} = \boldsymbol{R}_{j-1}^{-1} - \frac{\rho \boldsymbol{R}_{j-1}^{-1} \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{w}_{j-1} \boldsymbol{w}_{j-1}^{\mathrm{T}} \boldsymbol{R}_{\boldsymbol{y}} \boldsymbol{R}_{j-1}^{-1}}{1 + \rho \boldsymbol{w}_{j-1}^{\mathrm{T}} \boldsymbol{R}_{\boldsymbol{y}} \boldsymbol{R}_{j-1}^{-1} \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{w}_{j-1}}, \tag{8.117}$$

where $j = 2, \ldots, p$, $\rho \geq \lambda_N / \lambda_1$, $\boldsymbol{R}_1 = \boldsymbol{R}_{\boldsymbol{y}}$ and $\boldsymbol{w}_{j-1} = \boldsymbol{v}_{j-1}$ is the $(j-1)$th minor generalized eigenvector extracted. It holds that

$$\begin{aligned} \boldsymbol{R}_j \boldsymbol{v}_q &= (\boldsymbol{R}_{\boldsymbol{y}} + \rho \sum\nolimits_{i=1}^{j-1} \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} \boldsymbol{R}_{\boldsymbol{y}}) \boldsymbol{v}_q \\ &= \boldsymbol{R}_{\boldsymbol{y}} \boldsymbol{v}_q + \rho \sum\nolimits_{i=1}^{j-1} \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} \boldsymbol{R}_{\boldsymbol{y}} \boldsymbol{v}_q \\ &= \lambda_q \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{v}_q + \rho \lambda_q \sum\nolimits_{i=1}^{j-1} \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{v}_i \boldsymbol{v}_i^{\mathrm{T}} \boldsymbol{R}_{\boldsymbol{y}} \boldsymbol{v}_q \\ &= \begin{cases} (1+\rho) \lambda_q \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{v}_q & \text{for } q = 1, \ldots, j-1 \\ \lambda_q \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{v}_q & \text{for } q = j, \ldots, N \end{cases}. \end{aligned} \tag{8.118}$$

Thus, the matrix pencil $(\boldsymbol{R}_{\mathrm{j}}, \boldsymbol{R}_{\boldsymbol{x}})$ has eigenvalues $\lambda_j \leq \cdots \leq \lambda_N \leq (1+\rho) \lambda_1 \leq \cdots \leq (1+\rho) \lambda_{j-1}$ associated with eigenvectors $\boldsymbol{v}_j, \ldots, \boldsymbol{v}_N, \boldsymbol{v}_1 \ldots \boldsymbol{v}_{j-1}$. Equation (8.117) is obtained from (8.116) based on the SM-formula. That is to say, by replacing $\boldsymbol{R}_{\boldsymbol{y}}$ with $\boldsymbol{R}_j$ and $\boldsymbol{R}_{\boldsymbol{y}}^{-1}$ with $\boldsymbol{R}_j^{-1}$ in (8.111) and (8.113), we can estimate the jth minor generalized eigen pair $(\boldsymbol{v}_j, \lambda_j)$.

In the GPCA case, consider the following equation

$$\boldsymbol{R}_j = \boldsymbol{R}_{j-1} - \boldsymbol{R}_{\boldsymbol{x}} \boldsymbol{w}_{j-1} \boldsymbol{w}_{j-1}^{T} \boldsymbol{R}_{\boldsymbol{y}}, \tag{8.119}$$

where $\boldsymbol{R}_1 = \boldsymbol{R}_{\boldsymbol{y}}$, and $\boldsymbol{w}_{j-1} = \boldsymbol{v}_{N-j+1}$ is the $(j-1)$th principal generalized eigenvector extracted. By replacing $\boldsymbol{R}_{\boldsymbol{y}}$ with $\boldsymbol{R}_j$ in (8.114) and (8.115), we can estimate the jth principal generalized eigen pair $(\boldsymbol{v}_{N-j+1}, \lambda_{N-j+1})$.

8.4.2 Adaptive Implementation of Coupled Generalized Systems

In engineering practice, the matrices $\boldsymbol{R}_{\boldsymbol{y}}$ and $\boldsymbol{R}_{\boldsymbol{x}}$ are the covariance matrices of random input sequences $\{y(k)\}_{k \in z}$ and $\{x(k)\}_{k \in z}$, respectively. Thus, the matrix pencil $(\boldsymbol{R}_{\boldsymbol{y}}, \boldsymbol{R}_{\boldsymbol{x}})$ is usually unknown in advance, and even slowly changing over time if the signal is nonstationary. In that case, the matrices $\boldsymbol{R}_{\boldsymbol{y}}$ and $\boldsymbol{R}_{\boldsymbol{x}}$ are variables and

thus need to be estimated with online approach. In this section, we propose to update $\boldsymbol{R}_y$ and $\boldsymbol{R}_x$ with:

$$\widehat{\boldsymbol{R}}_y(k+1) = \beta\widehat{\boldsymbol{R}}_y(k) + \boldsymbol{y}(k+1)\boldsymbol{y}^H(k+1), \tag{8.120}$$

$$\widehat{\boldsymbol{R}}_x(k+1) = \alpha\widehat{\boldsymbol{R}}_x(k) + \boldsymbol{x}(k+1)\boldsymbol{x}^H(k+1). \tag{8.121}$$

By using the MS-formula, $\boldsymbol{Q}_y(k) = \widehat{\boldsymbol{R}}_y^{-1}(k)$ and $\boldsymbol{Q}_x(k) = \widehat{\boldsymbol{R}}_x^{-1}(k)$ can be updated as:

$$\boldsymbol{Q}_y(k+1) = \frac{1}{\beta}\left(\boldsymbol{Q}_y(k) - \frac{\boldsymbol{Q}_y(k)\boldsymbol{y}(k+1)\boldsymbol{y}^H(k+1)\boldsymbol{Q}_y(k)}{\alpha + \boldsymbol{y}^H(k+1)\boldsymbol{Q}_y(k)\boldsymbol{y}(k+1)}\right), \tag{8.122}$$

$$\boldsymbol{Q}_x(k+1) = \frac{1}{\alpha}\left(\boldsymbol{Q}_x(k) - \frac{\boldsymbol{Q}_x(k)\boldsymbol{x}(k+1)\boldsymbol{x}^H(k+1)\boldsymbol{Q}_x(k)}{\alpha + \boldsymbol{x}^H(k+1)\boldsymbol{Q}_x(k)\boldsymbol{x}(k+1)}\right). \tag{8.123}$$

It is known that

$$\lim_{k\to\infty}\frac{1}{k}\widehat{\boldsymbol{R}}_y(k) = \boldsymbol{R}_y \tag{8.124}$$

$$\lim_{k\to\infty}\frac{1}{k}\widehat{\boldsymbol{R}}_x(k) = \boldsymbol{R}_x \tag{8.125}$$

when $\alpha = \beta = 1$. By replacing $\boldsymbol{R}_y, \boldsymbol{R}_x, \boldsymbol{R}_y^{-1}$ and $\boldsymbol{R}_x^{-1}$ in (8.111)–(8.115) with $\widehat{\boldsymbol{R}}_y(k)$, $\widehat{\boldsymbol{R}}_x(k)$, $\boldsymbol{Q}_y(k)$ and $\boldsymbol{Q}_x(k)$, respectively, we can easily obtain the online GMCA algorithm with normalized step as:

$$\tilde{\boldsymbol{w}}(k+1) = \eta_1\frac{\boldsymbol{Q}_y(k+1)\widehat{\boldsymbol{R}}_x(k+1)\boldsymbol{w}(k)}{\boldsymbol{w}^H(k)\widehat{\boldsymbol{R}}_x(k+1)\boldsymbol{Q}_y(k+1)\widehat{\boldsymbol{R}}_x(k+1)\boldsymbol{w}(k)} + (1-\eta_1)\boldsymbol{w}(k), \tag{8.126}$$

$$\boldsymbol{w}(k+1) = \frac{\tilde{\boldsymbol{w}}(k+1)}{\|\tilde{\boldsymbol{w}}(k+1)\|_{\widehat{\boldsymbol{R}}_x(k+1)}}, \tag{8.127}$$

$$\lambda(k+1) = \gamma_1\frac{1}{\boldsymbol{w}^H(k)\widehat{\boldsymbol{R}}_x(k+1)\boldsymbol{Q}_y(k+1)\widehat{\boldsymbol{R}}_x(k+1)\boldsymbol{w}(k)} + (1-\gamma_1)\lambda(k), \tag{8.128}$$

and the online GPCA algorithm with normalized step as:

$$\tilde{\boldsymbol{w}}(k+1) = \eta_2 \frac{\boldsymbol{Q}_{\boldsymbol{x}}(k+1)\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k+1)\boldsymbol{w}(k)}{\boldsymbol{w}^H(k)\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k+1)\boldsymbol{Q}_{\boldsymbol{x}}(k+1)\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k+1)\boldsymbol{w}(k)} + (1-\eta_2)\boldsymbol{w}(k), \quad (8.129)$$

$$\boldsymbol{w}(k+1) = \frac{\tilde{\boldsymbol{w}}(k+1)}{\|\tilde{\boldsymbol{w}}(k+1)\|_{\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k+1)}}, \quad (8.130)$$

$$\lambda(k+1) = \gamma_2 \boldsymbol{w}^H(k)\,\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k+1)\boldsymbol{Q}_{\boldsymbol{x}}(k+1)\widehat{\boldsymbol{R}}_{\boldsymbol{y}}(k+1)\boldsymbol{w}\,(k) + (1-\gamma_2)\lambda(k), \quad (8.131)$$

where η_1, η_2, γ_1, $\gamma_2 \in (0, 1]$ are the step sizes.

In the rest of this section, for convenience, we refer to the GPCA and GMCA algorithms proposed in [4] as nGPCA and nGMCA for short, respectively, where *n* means that these algorithms were proposed by Nguyen. Similarly, we refer to the algorithm in (8.126)–(8.128) as fGMCA and the algorithm in (8.129)–(8.131) as fGPCA for short.

At the end of this section, we discuss the computational complexity of our algorithms. Taking fGMCA as an example, the computation of $\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k)$ and $\boldsymbol{Q}_{\boldsymbol{y}}(k)$ requires $5N^2 + O(N)$ multiplications. Moreover, by using (8.121), we have the following:

$$\begin{aligned}
&\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k+1)\boldsymbol{w}(k) \\
&= \left[\frac{k}{k+1}\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k) + \frac{1}{k+1}\boldsymbol{x}(k+1)\boldsymbol{x}^H(k+1)\right]\boldsymbol{w}(k) \\
&= \frac{k}{k+1}\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k)\boldsymbol{w}(k) + \frac{1}{k+1}\boldsymbol{x}(k+1)[\boldsymbol{x}^H(k+1)\boldsymbol{w}(k)],
\end{aligned} \quad (8.132)$$

where

$$\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k)\boldsymbol{w}(k) = \frac{\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k)\tilde{\boldsymbol{w}}(k)}{\sqrt{\tilde{\boldsymbol{w}}(k)^H\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k)\tilde{\boldsymbol{w}}(k)}}. \quad (8.133)$$

Since $\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k)\tilde{\boldsymbol{w}}(k)$ has been computed at the previous step when calculating the $\boldsymbol{R}_{\boldsymbol{x}}$-norm of $\boldsymbol{w}$(k), the update of $\widehat{\boldsymbol{R}}_{\boldsymbol{x}}(k+1)\boldsymbol{w}\,(k)$ requires only $O(N)$ multiplications. Thus, the updates of $\boldsymbol{w}(k)$ and $\lambda(k)$ in fGMCA requires $2N^2 + O(N)$ multiplications. Hence, fGMCA requires a total of $7N^2 + O(N)$ multiplications at each iteration. In a similar way, we can see that fGPCA also requires a total of $7N^2 + O(N)$ multiplications at each iteration. Thus, the computational complexity of both fGMCA and fGPCA is less than that of nGMCA and nGPCA (i.e., $10N^2 + O(N)$).

8.4.3 Convergence Analysis

The convergence of neural network learning algorithms is a difficult topic for direct study and analysis, and as pointed out [18], from the application point of view. The DDT method is more reasonable for studying the convergence of algorithms than traditional method. Using the DDT approach, Nguyen first reported the explicit convergence analysis of coupled generalized eigen pair extraction algorithms [4]. In this section, we will also analyze the convergence of our algorithms with the DDT approach on the basis of [4].

The DDT system of fGMCA is given as:

$$\tilde{\boldsymbol{w}}(k+1) = \boldsymbol{w}(k) + \eta_1 \left[\frac{\boldsymbol{Q}_y \widehat{\boldsymbol{R}}_x \boldsymbol{w}(k)}{\boldsymbol{w}^H(k) \widehat{\boldsymbol{R}}_x \boldsymbol{Q}_y \widehat{\boldsymbol{R}}_x \boldsymbol{w}(k)} - \boldsymbol{w}(k) \right], \tag{8.134}$$

$$\boldsymbol{w}(k+1) = \frac{\tilde{\boldsymbol{w}}(k+1)}{\|\tilde{\boldsymbol{w}}(k+1)\|_{\boldsymbol{R}_x}}, \tag{8.135}$$

$$\lambda(k+1) = \lambda(k) + \gamma_1 \left[\frac{1}{\boldsymbol{w}^H(k) \widehat{\boldsymbol{R}}_x \boldsymbol{Q}_y \widehat{\boldsymbol{R}}_x \boldsymbol{w}(k)} - \lambda(k) \right]. \tag{8.136}$$

which is referred to as DDT System 1.

And the DDT system of fGPCA is given as:

$$\tilde{\boldsymbol{w}}(k+1) = \boldsymbol{w}(k) + \eta_2 \left[\frac{\boldsymbol{Q}_x \widehat{\boldsymbol{R}}_y \boldsymbol{w}(k)}{\boldsymbol{w}^H(k) \widehat{\boldsymbol{R}}_y \boldsymbol{Q}_x \widehat{\boldsymbol{R}}_y \boldsymbol{w}(k)} - \boldsymbol{w}(k) \right], \tag{8.137}$$

$$\boldsymbol{w}(k+1) = \frac{\tilde{\boldsymbol{w}}(k+1)}{\|\tilde{\boldsymbol{w}}(k+1)\|_{\boldsymbol{R}_y}}, \tag{8.138}$$

$$\lambda(k+1) = \lambda(k) + \gamma_2 [\boldsymbol{w}^H(k) \widehat{\boldsymbol{R}}_y \boldsymbol{Q}_x \widehat{\boldsymbol{R}}_y \boldsymbol{w}(k) - \lambda(k)]. \tag{8.139}$$

which is referred to as DDT System 2.

Similar to [4], we also denote by $\|\boldsymbol{u}\|_{\boldsymbol{R}} = \sqrt{\boldsymbol{u}^H \boldsymbol{R} \boldsymbol{u}}$ the $\boldsymbol{R}$-norm of a vector $\boldsymbol{u}$, where $\boldsymbol{R} \in C^{N \times N}$ and $\boldsymbol{u} \in C^N$, $\boldsymbol{P}_{\boldsymbol{V}}^{\boldsymbol{R}}(\boldsymbol{u}) \in \boldsymbol{V}$ is the $\boldsymbol{R}$-orthogonal projection of $\boldsymbol{u}$ onto a subspace $\boldsymbol{V} \in C^N$; i.e., $\boldsymbol{P}_{\boldsymbol{V}}^{\boldsymbol{R}}(\boldsymbol{u})$ is the unique vector satisfying $\left\|\boldsymbol{u} - \boldsymbol{P}_{\boldsymbol{V}}^{\boldsymbol{R}}(\boldsymbol{u})\right\|_{\boldsymbol{R}} = \min_{\boldsymbol{v} \in \boldsymbol{V}} \|\boldsymbol{u} - \boldsymbol{v}\|_{\boldsymbol{R}}$, $\boldsymbol{V}_{\lambda_i}$ is the generalized eigen-subspace associated with the ith smallest generalized eigenvalue λ_i, i.e., $\boldsymbol{V}_{\lambda_i} = \left\{\boldsymbol{v} \in C^N | \boldsymbol{R}_y \boldsymbol{v} = \lambda_i \boldsymbol{R}_x \boldsymbol{v}\right\}$ $(i = 1, 2, \ldots, N)$. (Note that $\boldsymbol{V}_{\lambda_i} = \boldsymbol{V}_{\lambda_j}$ if $\lambda_i = \lambda_j$ for some $i \neq j$), $\boldsymbol{V}_{<\boldsymbol{R}>}^{\perp}$ is the $\boldsymbol{R}$-orthogonal complement subspace of $\boldsymbol{V}$ for any subspace $\boldsymbol{V} \subset C^N$, *i.e.*, $\boldsymbol{V}_{<\boldsymbol{R}>}^{\perp} = \{\boldsymbol{u} \in C^N \mid <\boldsymbol{u}, \boldsymbol{v}>_{\boldsymbol{R}} = \boldsymbol{v}^H \boldsymbol{R} \boldsymbol{u} = 0, \ \forall \boldsymbol{v} \in \boldsymbol{V}\}$.

Next, we will present two theorems to show the convergence of our algorithms. In the following, two cases will be considered. In Case 1, $\lambda_1 = \lambda_2 = \cdots = \lambda_N$ and in Case 2, $\lambda_1 < \lambda_N$.

Theorem 8.1 (Convergence analysis of fGMCA) *Suppose that the sequence* $[\boldsymbol{w}(k), \lambda(k)]_{k=0}^{\infty}$ *is generated by DDT System 1 with any* $\eta_1, \gamma_1 \in (0, 1]$, *any initial* $\boldsymbol{R}_x$*-normalized vector* $\boldsymbol{w}(0) \notin (\boldsymbol{V}_{\lambda_1})^{\perp}_{<\boldsymbol{R}_x>}$, *and any* $\lambda(0) > 0$. *Then for Case 1, it holds that* $\boldsymbol{w}(k) = \boldsymbol{w}(0)$ *for all* $k \geq 0$, *which is also a generalized eigenvector associated with the generalized eigenvalue* λ_1 *of the matrix pencil* $(\boldsymbol{R}_y, \boldsymbol{R}_x)$, *and* $\lim_{k\to\infty} \lambda(k) = \lambda_1$. *For Case 2, it holds that*

$$\lim_{k\to\infty} \boldsymbol{w}(k) = \frac{\boldsymbol{P}^{\boldsymbol{R}_x}_{\boldsymbol{V}_{\lambda_1}}[\boldsymbol{w}(0)]}{\left\|\boldsymbol{P}^{\boldsymbol{R}_x}_{\boldsymbol{V}_{\lambda_1}}[\boldsymbol{w}(0)]\right\|_{\boldsymbol{R}_x}}, \tag{8.140}$$

$$\lim_{k\to\infty} \lambda(k) = \lambda_1. \tag{8.141}$$

Proof Case 1:

Since $\lambda_1 = \lambda_2 = \cdots = \lambda_N$ ensures $\boldsymbol{V}_{\lambda_1} = C^N$, we can verify that for all $k \geq 0$ that $\boldsymbol{w}(k) = \boldsymbol{w}(0) \neq \boldsymbol{0}$, which is also a generalized eigenvector associated with the generalized eigenvalue λ_1 of matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_y)$. Moreover, from (8.128) we have $\lambda(k+1) = (1-\gamma_1)\lambda(k) + \gamma_1\lambda_1$ for all $k \geq 0$. Hence

$$\begin{aligned}
\lambda(k+1) &= (1-\gamma_1)\lambda(k) + \gamma_1\lambda_1 = \cdots \\
&= (1-\gamma_1)^{k+1}\lambda(0) + \gamma_1\lambda_1[1 + (1-\gamma_1) + \cdots + (1-\gamma_1)^k] \\
&= (1-\gamma_1)^{k+1}\lambda(0) + \lambda_1[1 - (1-\gamma_1)^{k+1}] \\
&= \lambda_1 + (1-\gamma_1)^{k+1}[\lambda(0) - \lambda_1].
\end{aligned} \tag{8.142}$$

Since $\gamma_1 \in (0, 1]$, we can verify that $\lim_{k\to\infty} \lambda(k) = \lambda_1$.

Case 2: Suppose that the generalized eigenvalues of the matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ have been ordered as $\lambda_1 = \cdots = \lambda_r < \lambda_{r+1} \leq \cdots \leq \lambda_N$ $(1 \leq r \leq N)$. Since $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_N\}$ is an $\boldsymbol{R}_x$-orthonormal basis of C^N, $\boldsymbol{w}(k)$ in DDT System 1 can be written uniquely as:

$$\boldsymbol{w}(k) = \sum_{i=1}^{N} z_i(k)\,\boldsymbol{v}_i,\ k = 0, 1, \ldots \tag{8.143}$$

where $z_i(k) = \langle \boldsymbol{w}(k), \boldsymbol{v}_i \rangle_{\boldsymbol{R}_x} = \boldsymbol{v}_i^H \boldsymbol{R}_x \boldsymbol{w}(k)$, $i = 1, 2, \ldots, N$.

First, we will prove by mathematical induction that for all $k > 0$, $\boldsymbol{w}(k)$ is well defined, $\boldsymbol{R}_x$-normalized, i.e.,

$$\boldsymbol{w}(k)^H \boldsymbol{R}_x \boldsymbol{w}(k) = \sum_{i=1}^{N} |z_i(k)|^2 = 1, \tag{8.144}$$

and $\boldsymbol{w}(k) \notin (\boldsymbol{V}_{\lambda_1})^{\perp}_{\langle \boldsymbol{R}_x \rangle}$, i.e., $[z_1(k), z_2(k), \ldots, z_r(k)] \neq 0$. Note that $\boldsymbol{w}(0) \notin (\boldsymbol{V}_{\lambda_1})^{\perp}_{\langle \boldsymbol{R}_x \rangle}$ is $\boldsymbol{R}_x$-normalized. Assume that $\boldsymbol{w}(k)$ is well defined, $\boldsymbol{R}_x$-normalized, and $\boldsymbol{w}(k) \notin (\boldsymbol{V}_{\lambda_1})^{\perp}_{\langle \boldsymbol{R}_x \rangle}$ for some $k > 0$. By letting $\tilde{\boldsymbol{w}}(k+1) = \sum_{i=1}^{N} \tilde{z}_i(k+1)\mathbf{v}_i$, from (8.134) and (8.143), we have the following:

$$\tilde{z}_i(k+1) = z_i(k)\left\{1 + \eta_1 \left[\frac{1}{\lambda_i \boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)} - 1\right]\right\}. \tag{8.145}$$

Since matrix pencil $\left(\boldsymbol{R}_x, \boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\right)$ has the same eigen pairs as $\left(\boldsymbol{R}_y, \boldsymbol{R}_x\right)$, and $\boldsymbol{w}(k)$ is $\boldsymbol{R}_x$-normalized, it follows that

$$\lambda_1 \leq \frac{\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{w}(k)}{\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)} = \frac{1}{\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)} \leq \lambda_N, \tag{8.146}$$

which is a generalization of the Rayleigh–Ritz ratio [19]. For $i = 1, \ldots, r$, (8.146) and (8.145) guarantee that

$$1 + \eta_1 \left[\frac{1}{\lambda_i \boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)} - 1\right] = 1 + \eta_1 \left[\frac{1}{\lambda_1}\frac{1}{\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)} - 1\right] \geq 1, \tag{8.147}$$

and $[z_1(k+1), z_2(k+1), \ldots, z_r(k+1)] \neq \mathbf{0}$. These imply that $\tilde{\boldsymbol{w}}(k+1) \neq 0$ and $\boldsymbol{w}(k+1) = \sum_{i=1}^{N} z_i(k+1)\,\boldsymbol{v}_i$ is well defined, $\boldsymbol{R}_x$-normalized, and $\boldsymbol{w}(k+1) \notin (\boldsymbol{V}_{\lambda_1})^{\perp}_{\langle \boldsymbol{R}_x \rangle}$, where

$$z_i(k+1) = \frac{\tilde{z}_i(k+1)}{\|\tilde{\boldsymbol{w}}(k+1)\|_{\boldsymbol{R}_x}}. \tag{8.148}$$

Therefore, $\boldsymbol{w}(k)$ is well defined, $\boldsymbol{R}_x$-normalized, and $\boldsymbol{w}(k) \notin (\boldsymbol{V}_{\lambda_1})^{\perp}_{\langle \boldsymbol{R}_x \rangle}$ for all $k \geq 0$.

Second, we will prove (8.125). Note that $\boldsymbol{w}(0) \notin (\boldsymbol{V}_{\lambda_1})^{\perp}_{\langle \boldsymbol{R}_x \rangle}$ implies the existence of some $m \in \{1, \ldots, r\}$ satisfying $z_m(0) \neq 0$, where $\lambda_1 = \cdots = \lambda_m = \cdots = \lambda_r$. From (8.145) and (8.148), we have $z_m(k+1)/z_m(0) > 0$ for all $k \geq 0$. By using (8.145) and (8.148), we can see that for $i = 1, \ldots, r$, it holds that

$$\begin{aligned}\frac{z_i(k+1)}{z_m(k+1)} &= \frac{\tilde{z}_i(k+1)}{\|\tilde{\boldsymbol{w}}(k+1)\|_{\boldsymbol{R}_x}} \frac{\|\tilde{\boldsymbol{w}}(k+1)\|_{\boldsymbol{R}_x}}{\tilde{z}_m(k+1)} \\ &= \frac{z_i(k)}{z_m(k)} \cdot \frac{1+\eta_1\left[\frac{1}{\lambda_i \boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)}-1\right]}{1+\eta_1\left[\frac{1}{\lambda_m \boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)}-1\right]} \\ &= \frac{z_i(k)}{z_m(k)} = \cdots = \frac{z_i(0)}{z_m(0)}.\end{aligned} \tag{8.149}$$

On the other hand, by using (8.145) and (8.148), we have for all $k \geq 0$ and $i = r+1, \ldots, N$ that

$$\begin{aligned}\frac{|z_i(k+1)|^2}{|z_m(k+1)|^2} &= \frac{\tilde{z}_i(k+1)}{\|\tilde{\boldsymbol{w}}(k+1)\|_{\boldsymbol{R}_x}} \frac{\|\tilde{\boldsymbol{w}}(k+1)\|_{\boldsymbol{R}_x}}{\tilde{z}_m(k+1)} \\ &= \left[\frac{1+\eta_1\left(\frac{1}{\lambda_i \boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)}-1\right)}{1+\eta_1\left(\frac{1}{\lambda_m \boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)}-1\right)}\right]^2 \cdot \frac{|z_i(k)|^2}{|z_m(k)|^2} \\ &= \left[1-\frac{\frac{1}{\lambda_1}-\frac{1}{\lambda_i}}{(\frac{1}{\eta_1}-1)\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)+\frac{1}{\lambda_1}}\right]^2 \cdot \frac{|z_i(k)|^2}{|z_m(k)|^2} = \psi(k)\frac{|z_i(k)|^2}{|z_m(k)|^2},\end{aligned} \tag{8.150}$$

where

$$\psi(k) = \left[1-\frac{\frac{1}{\lambda_1}-\frac{1}{\lambda_i}}{\left(\frac{1}{\eta_1}-1\right)\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)+\frac{1}{\lambda_1}}\right]^2. \tag{8.151}$$

For all $i = r+1, \ldots, N$, together with $\eta_1 \in (0, 1]$ and $1/\lambda_1 - 1/\lambda_i > 0$, Eq. (8.146) guarantees that

$$\begin{aligned}1-\frac{\frac{1}{\lambda_1}-\frac{1}{\lambda_i}}{\left(\frac{1}{\eta_1}-1\right)\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)+\frac{1}{\lambda_1}} &\leq 1-\frac{\frac{1}{\lambda_1}-\frac{1}{\lambda_{r+1}}}{\left(\frac{1}{\eta_1}-1\right)\frac{1}{\lambda_1}+\frac{1}{\lambda_1}} \\ &= 1-\eta_1\left(1-\frac{\lambda_1}{\lambda_{r+1}}\right) < 1,\end{aligned} \tag{8.152}$$

and

$$1-\frac{\frac{1}{\lambda_1}-\frac{1}{\lambda_i}}{\left(\frac{1}{\eta_1}-1\right)\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)+\frac{1}{\lambda_1}}\geq 1-\frac{\frac{1}{\lambda_1}-\frac{1}{\lambda_N}}{\left(\frac{1}{\eta_1}-1\right)\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)+\frac{1}{\lambda_1}}$$
$$=1-\frac{\frac{1}{\lambda_1}-\frac{1}{\lambda_N}}{\frac{1}{\eta_1}\frac{1}{\lambda_N}+\left(\frac{1}{\lambda_1}-\frac{1}{\lambda_N}\right)}>0. \tag{8.153}$$

From (8.152) and (8.153), we can verify that

$$0<\psi(k)<1,\ i=r+1,\ldots,N, \tag{8.154}$$

for all $k \geq 0$. Denote $\psi_{\max} = \max\{\psi(k)|k \geq 0\}$. Clearly $0 < \psi_{\max} < 1$. From (8.150), we have the following:

$$\frac{|z_i(k+1)|^2}{|z_m(k+1)|^2}\leq\psi_{\max}\frac{|z_i(k)|^2}{|z_m(k)|^2}\leq\cdots\leq\psi_{\max}^{k+1}\frac{|z_i(0)|^2}{|z_m(0)|^2}. \tag{8.155}$$

Since $\boldsymbol{w}(k)$ is $\boldsymbol{R}_x$-normalized, $|z_m(k)|^2 \leq 1$ for all k ≥ 0, it follows from (8.155) that

$$\begin{aligned}\sum_{i=r+1}^{N}|z_i(k)|^2&\leq\sum_{i=r+1}^{N}\frac{|z_i(k)|^2}{|z_m(k)|^2}\leq\cdots\\&\leq\psi_{\max}^k\sum_{i=r+1}^{N}\frac{|z_i(0)|^2}{|z_m(0)|^2}\to 0\text{ as }k\to\infty,\end{aligned} \tag{8.156}$$

which along with (8.144) implies that

$$\lim_{k\to\infty}\sum_{i=1}^{r}|z_i(k)|^2=1. \tag{8.157}$$

Note that $z_m(k)/z_m(0) > 0$ for all $k \geq 0$. Then, from (8.149) and (8.157) we have the following:

$$\lim_{k\to\infty}z_i(k)=\frac{z_i(0)}{\sqrt{\sum_{j=1}^{r}|z_j(0)|^2}},\ i=1,2,\ldots,r. \tag{8.158}$$

Based on (8.156) and (8.158), (8.140) can be obtained as follows:

$$\lim_{k\to\infty}\boldsymbol{w}(k)=\sum_{i=1}^{r}\frac{z_i(0)}{\sqrt{\sum_{j=1}^{r}|z_j(0)|^2}}\boldsymbol{v}_i=\frac{\boldsymbol{P}_{\boldsymbol{V}_{\lambda_1}}^{\boldsymbol{R}_x}[\boldsymbol{w}(0)]}{\left\|\boldsymbol{P}_{\boldsymbol{V}_{\lambda_1}}^{\boldsymbol{R}_x}[\boldsymbol{w}(0)]\right\|_{\boldsymbol{R}_x}}. \tag{8.159}$$

Finally, we will prove (8.141). From (8.159), we can see that

$$\lim_{k\to\infty}\frac{1}{\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)}=\lambda_1. \tag{8.160}$$

That is, for any small positive δ, there exists a $K>0$ satisfying

$$\lambda_1-\delta<\frac{1}{\boldsymbol{w}^H(k)\boldsymbol{R}_x\boldsymbol{R}_y^{-1}\boldsymbol{R}_x\boldsymbol{w}(k)}<\lambda_1+\delta, \tag{8.161}$$

for all $k>K$. It follows from (8.128) that

$$\begin{aligned}\lambda(k)&>(1-\gamma_1)\lambda(k-1)+\gamma_1(\lambda_1-\delta)>\cdots>(1-\gamma_1)^{k-K}\lambda(K)+\gamma_1(\lambda_1-\delta)\\&\times\left[1+(1-\gamma_1)+\cdots+(1-\gamma_1)^{k-K}\right]=(1-\gamma_1)^{k-K}\lambda(K)+\gamma_1(\lambda_1-\delta)\\&\times\left[1-(1-\gamma_1)^{k-K}\right]=(\lambda_1-\delta)+(1-\gamma_1)^{k-K}[\lambda(K)-\lambda_1+\delta],\end{aligned} \tag{8.162}$$

and

$$\begin{aligned}\lambda(k)&<(1-\gamma_1)\lambda(k-1)+\gamma_1(\lambda_1+\delta)<\cdots<(1-\gamma_1)^{k-K}\lambda(K)+\gamma_1(\lambda_1+\delta)\\&\times\left[1+(1-\gamma_1)+\cdots+(1-\gamma_1)^{k-K-1}\right]=(1-\gamma_1)^{k-K}\lambda(K)+(\lambda_1+\delta)\\&\times\left[1-(1-\gamma_1)^{k-K}\right]=(\lambda_1+\delta)+(1-\gamma_1)^{k-K}[\lambda(K)-\lambda_1-\delta],\end{aligned} \tag{8.163}$$

for all $k>K$. Since $\gamma_1\in(0,1]$, it is easy to verify from (8.162) and (8.163) that $\lim_{k\to\infty}\lambda(k)=\lambda_1$.

This completes the proof.

Theorem 8.2 (Convergence analysis of fGPCA) *Suppose that the sequence* $[\boldsymbol{w}(k),\lambda(k)]_{k=0}^{\infty}$ *is generated by DDT System 2 with any* $\eta_2,\gamma_2\in(0,1]$, *any initial* $\boldsymbol{R}_y$*-normalized vector* $\boldsymbol{w}(0)\notin(\boldsymbol{V}_{\lambda_N})^{\perp}_{<\boldsymbol{R}_x>}$, *and any* $\lambda(0)>0$. *Then for Case 1, it holds that* $\boldsymbol{w}(k)=\boldsymbol{w}(0)$ *for all* $k\geq 0$, *which is also a generalized eigenvector associated with the generalized eigenvalue* λ_N *of the matrix pencil* $(\boldsymbol{R}_y,\boldsymbol{R}_x)$, *and* $\lim_{k\to\infty}\lambda(k)=\lambda_N$. *For Case 2, it holds that*

$$\lim_{k\to\infty}\boldsymbol{w}(k)=\sqrt{\frac{1}{\lambda_N}}\frac{\boldsymbol{P}^{\boldsymbol{R}_x}_{\boldsymbol{V}_{\lambda_N}}[\boldsymbol{w}(0)]}{\left\|\boldsymbol{P}^{\boldsymbol{R}_x}_{\boldsymbol{V}_{\lambda_N}}[\boldsymbol{w}(0)]\right\|_{\boldsymbol{R}_x}}, \tag{8.164}$$

$$\lim_{k\to\infty} \lambda(k) = \lambda_N. \tag{8.165}$$

The proof of Theorem 8.2 is similar to that of Theorem 8.1. A minor difference is that we need to calculate the $\boldsymbol{R}_y$-norm of $\boldsymbol{w}(k)$ at each step. Another minor difference is that in (8.146), it holds that matrix pencil $(\boldsymbol{R}_y\boldsymbol{R}_x^{-1}\boldsymbol{R}_y, \boldsymbol{R}_y)$ has the same eigen pairs as $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ and $\boldsymbol{w}(k)$ is well defined, $\boldsymbol{R}_y$-normalized, and $\boldsymbol{w}(k) \notin (\boldsymbol{V}_{\lambda_N})^{\perp}_{\langle \boldsymbol{R}_x \rangle}$ for all $k \geq 0$. Therefore,

$$\lambda_1 \leq \frac{\boldsymbol{w}^H(k)\boldsymbol{R}_y\boldsymbol{R}_x^{-1}\boldsymbol{R}_y\boldsymbol{w}(k)}{\boldsymbol{w}^H(k)\boldsymbol{R}_y\boldsymbol{w}(k)} = \boldsymbol{w}^H(k)\boldsymbol{R}_y\boldsymbol{R}_x^{-1}\boldsymbol{R}_y\boldsymbol{w}(k) \leq \lambda_N. \tag{8.166}$$

Particularly, if λ_1 and λ_2 are distinct $(\lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N)$, we have $\boldsymbol{V}_{\lambda_1} = \text{span}\{\boldsymbol{V}_1\}$, $\boldsymbol{P}^{\boldsymbol{R}_x}_{\boldsymbol{V}_{\lambda_1}}[\boldsymbol{w}(0)] = \langle \boldsymbol{w}(0), \boldsymbol{V}_1 \rangle_{\boldsymbol{R}_x} \boldsymbol{V}_1$, and $\left\| \boldsymbol{P}^{\boldsymbol{R}_x}_{\boldsymbol{V}_{\lambda_1}}[\boldsymbol{w}(0)] \right\|_{\boldsymbol{R}_x} = \left| \langle \boldsymbol{w}(0), \boldsymbol{V}_1 \rangle_{\boldsymbol{R}_x} \right|$. Moreover, if λ_{N-1} and λ_N are distinct $(\lambda_1 \leq \cdots \leq \lambda_{N-1} < \lambda_N)$, we have $\boldsymbol{V}_{\lambda_N} = \text{span}\{\boldsymbol{V}_N\}$, $\boldsymbol{P}^{\boldsymbol{R}_y}_{\boldsymbol{V}_{\lambda_N}}[\boldsymbol{w}(0)] = \langle \boldsymbol{w}(0), \boldsymbol{V}_N \rangle_{\boldsymbol{R}_y} \boldsymbol{V}_N$ and $\left\| \boldsymbol{P}^{\boldsymbol{R}_y}_{\boldsymbol{V}_{\lambda_N}}[\boldsymbol{w}(0)] \right\|_{\boldsymbol{R}_y} = \left| \langle \boldsymbol{w}(0), \boldsymbol{V}_N \rangle_{\boldsymbol{R}_y} \right|$. Hence, the following corollaries hold.

Corollary 8.1 *Suppose that $\lambda_1 < \lambda_2 \leq \cdots \leq \lambda_N$. Then the sequence $[\boldsymbol{w}(k), \lambda(k)]_{k=0}^{\infty}$ generated by DDT System 1 with any $\eta_1, \gamma_1 \in (0, 1]$, any initial $\boldsymbol{R}_x$-normalized vector $\boldsymbol{w}(0) \notin (\boldsymbol{V}_{\lambda_1})^{\perp}_{\langle \boldsymbol{R}_x \rangle}$, and any $\lambda(0) > 0$ satisfies*

$$\lim_{k\to\infty} \boldsymbol{w}(k) = \frac{\langle \boldsymbol{w}(0), \boldsymbol{V}_1 \rangle_{\boldsymbol{R}_x} \boldsymbol{V}_1}{\left| \langle \boldsymbol{w}(0), \boldsymbol{V}_1 \rangle_{\mathbf{R}_x} \right|}, \tag{8.167}$$

$$\lim_{k\to\infty} \lambda(k) = \lambda_1. \tag{8.168}$$

Corollary 8.2 *Suppose that $\lambda_1 \leq \cdots \leq \lambda_{N-1} < \lambda_N$. Then the sequence $[\boldsymbol{w}(k), \lambda(k)]_{k=0}^{\infty}$ generated by DDT System 2 with any $\eta_2, \gamma_2 \in (0, 1]$, any initial $\boldsymbol{R}_y$-normalized vector $\boldsymbol{w}(0) \notin (\boldsymbol{V}_{\lambda_N})^{\perp}_{\langle \boldsymbol{R}_y \rangle}$, and any $\lambda(0) > 0$ satisfies*

$$\lim_{k\to\infty} \boldsymbol{w}(k) = \sqrt{\frac{1}{\lambda_N}} \frac{\langle \boldsymbol{w}(0), \boldsymbol{V}_N \rangle_{\boldsymbol{R}_x} \boldsymbol{V}_N}{\left| \langle \boldsymbol{w}(0), \boldsymbol{V}_N \rangle_{\boldsymbol{R}_x} \right|}, \tag{8.169}$$

$$\lim_{k\to\infty} \lambda(k) = \lambda_N. \tag{8.170}$$

8.4.4 Numerical Examples

In this section, we present two numerical examples to evaluate the performance of our algorithms (fGMCA and fGPCA). The first estimates the principal and minor

generalized eigenvectors from two random vector processes, which are generated by two sinusoids with additive noise. The second illustrates performance of our algorithms for the BSS problem. Besides nGMCA and nGPCA, we also compare with the following algorithms, which were proposed in the recent ten years:

(1) Gradient-based: adaptive version of ([4], Alg. 2) with negative (for GPCA) and positive (for GMCA) step sizes;
(2) Power-like: fast generalized eigenvector tracking [20] based on the power method;
(3) R-GEVE: reduced-rank generalized eigenvector extraction algorithm [21];
(4) Newton-type: adaptive version of Alg. I proposed in [22].

A. *Experiment 1*

In this experiment, the input samples are generated by:

$$y(n) = \sqrt{2}\,\sin(0.62\pi n + \theta_1) + \varsigma_1(n), \tag{8.171}$$

$$x(n) = \sqrt{2}\sin(0.46\pi n + \theta_2) + \sqrt{2}\sin(0.74\pi n + \theta_3) + \varsigma_2(n), \tag{8.172}$$

where θ_i (i = 1, 2, 3) are the initial phases, which follow uniform distributions within [0, 2π], and $\zeta_1(n)$ and $\zeta_2(n)$ are zero-mean white noises with variance $\sigma_1^2 = \sigma_2^2 = 0.1$.

The input vectors $\{\boldsymbol{y}(k)\}$ and $\{\boldsymbol{x}(k)\}$ are arranged in blocks of size N = 8, i.e., $\boldsymbol{y}(k) = [y(k),\ldots, y(k - N+1)]^{\mathrm{T}}$ and $\boldsymbol{x}(k) = [x(k),\ldots, x(k - N + 1)]^{\mathrm{T}}$, $k \geq N$. Define the $N \times N$ matrix pencil $(\overline{\boldsymbol{R}}_{\boldsymbol{y}}, \overline{\boldsymbol{R}}_{\boldsymbol{x}})$ with the (p, q) entry (p, q = 1,2,…,N) of $\overline{\boldsymbol{R}}_{\boldsymbol{y}}$ and $\overline{\boldsymbol{R}}_{\boldsymbol{x}}$ given by

$$\left[\overline{\boldsymbol{R}}_{\boldsymbol{y}}\right]_{pq} = \cos[0.62\pi\,(p-q)] + \delta_{pq}\sigma_1^2, \tag{8.173}$$

$$\left[\overline{\boldsymbol{R}}_{\boldsymbol{x}}\right]_{pq} = \cos[0.46\pi\,(p-q)] + \cos[0.74\pi\,(p-q)] + \delta_{pq}\sigma_2^2. \tag{8.174}$$

For comparison, the direction cosine DC(k) is used to measure the accuracy of direction estimate. We also measure the numerical stability of all algorithms by the sample standard deviation of the direction cosine:

$$SSD(k) = \sqrt{\frac{1}{L-1}\sum_{j=1}^{L}\left[\mathrm{DC}_j(k) - \overline{\mathrm{DC}}(k)\right]^2}, \tag{8.175}$$

where $\mathrm{DC}_j(k)$ is the direction cosine of the jth independent run (j = 1, 2,…, L) and $\overline{\mathrm{DC}}(k)$ is the average over L = 100 independent runs.

In this example, we conduct two simulations. In the first simulation, we use fGMCA, nGMCA, and the other aforementioned algorithms to extract the minor

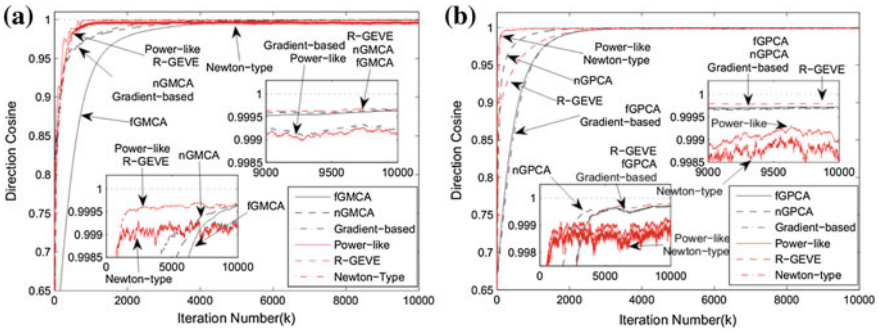

Fig. 8.10 Example 1: Direction cosine of the principal/minor generalized eigenvector. **a** First simulation. **b** Second simulation

generalized eigenvector of matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$. Note that in gradient-based algorithm a positive step size is used, and the other algorithms are applied to estimate the principal generalized eigenvector of matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$ which is also the minor generalized eigenvector of $(\boldsymbol{R}_y, \boldsymbol{R}_x)$. In the second simulation, we use fGPCA, nGPCA, and the other algorithms to extract the principal generalized eigenvector of matrix pencil $(\boldsymbol{R}_y, \boldsymbol{R}_x)$. Note that in gradient-based algorithm a negative step size is used. The sets of parameters used in simulations refer to [4], [22]. All algorithms have been initialized with $\widehat{\boldsymbol{R}}_x(0) = \widehat{\boldsymbol{R}}_y(0) = \boldsymbol{Q}_x(0) = \boldsymbol{Q}_y(0) = \boldsymbol{I}_N$ (if used) and $\boldsymbol{w}(0) = \boldsymbol{e}_1$, where $\boldsymbol{e}_1$ stands for the first columns of $\boldsymbol{I}_N$.

The experimental results are shown in Figs. 8.10 to 8.12 and Table 8.1.

Figures 8.10 and 8.11 depict the time course of direction cosine for generalized eigenvector estimation and sample standard deviation of the direction cosine. The results of minor and principal generalized eigenvalues estimation of all generalized eigen-pair extraction algorithms are shown in Fig. 8.12. We find that fGM(P)CA converge faster than nGMCA and nGPCA at the beginning steps, respectively, and

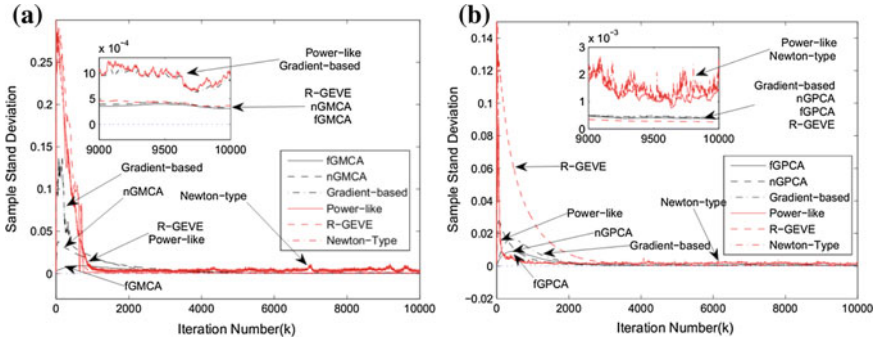

Fig. 8.11 Example 1: Sample standard deviation of the direction cosine. **a** First simulation. **b** Second simulation

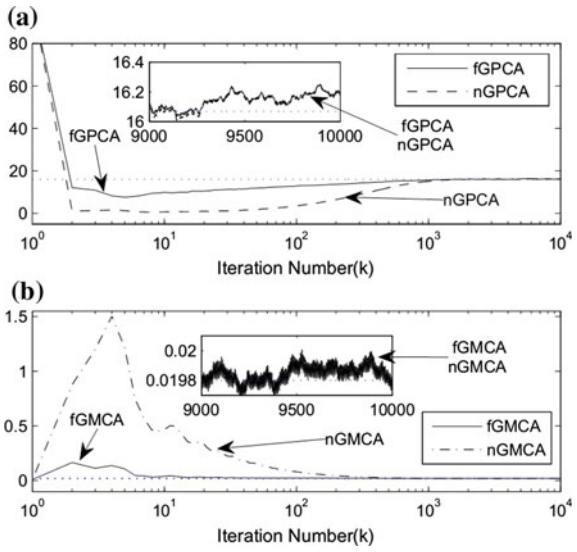

Fig. 8.12 Example 1: Generalized eigenvalues estimation. **a** First simulation: principal generalized eigenvalues estimation. **b** Second simulation: minor generalized eigenvalues estimation

Table 8.1 Computational complexity of all algorithms

Algorithm	fGM(P)CA	nGM(P)CA	Gradient-based
Complexity	$7N^2 + O(N)$	$10N^2 + O(N)$	$10N^2 + O(N)$
Algorithm	Power-like	R-GEVE	Newton-type
Complexity	$13N^2 + O(N)$	$6N^2 + O(N)$	$4N^2 + O(N)$

fGMCA and fGPCA have similar estimation accuracy as nGMCA and nGPCA, respectively. Figure 8.12 shows that all generalized eigen-pair extraction algorithms can extract the principal or minor generalized eigenvalue efficiently.

The computational complexities of all aforementioned algorithms are shown in Table 8.1. We find that Newton-type has the lowest computational complexity but the worst estimation accuracy and standard deviation. The Power-like has the highest computational complexity compared with the other algorithms. The nGM(P)CA and gradient-based algorithms have same computational complexity. The computational complexities of R-GEVE and the proposed algorithms are similar, which are lower than that of nGM(P)CA and gradient-based algorithms.

B. *Experiment 2*

We perform this experiment to show the performance of our algorithm for the BSS problem. Consider a linear BSS model [23]:

$$\boldsymbol{x}(n) = \boldsymbol{A}\boldsymbol{s}(n) + \boldsymbol{e}(n), \tag{8.176}$$

where $\boldsymbol{x}(n)$ is a r-dimensional vector of the observed signals at time k, $\boldsymbol{s}(n)$ is a l-dimensional vector of the unknown source signals, $\boldsymbol{A} \in \boldsymbol{R}^{l\times r}$ denotes the unknown

mixing matrix, and $\boldsymbol{e}(n)$ is an unknown noise vector. In general, BSS problem is that of finding a separating matrix $\boldsymbol{W}$ such that the r-dimensional output signal vector $\boldsymbol{y} = \boldsymbol{W}^{\mathrm{T}}\boldsymbol{x}$ contains components that are as independent as possible. In this experiment, we compare the proposed algorithms with nGMCA and nGPCA algorithms, as well as batch-processing generalized eigenvalue decomposition method (EVD method in MATLAB software). We use the method given in [20, 22] to formulate the matrix pencil by applying FIR filtering. $z(n)$, the output of FIR filter, is given as

$$\boldsymbol{z}(n) = \sum_{t=0}^{m} \tau(t)\boldsymbol{x}(n-t), \tag{8.177}$$

where $\tau(t)$ are the coefficients of the FIR filter. Let $\boldsymbol{R}_{\boldsymbol{x}} = E[\boldsymbol{x}(k)\boldsymbol{x}^{\mathrm{T}}(k)]$ and $\boldsymbol{R}_{z} = E[\boldsymbol{z}(k)\boldsymbol{z}^{\mathrm{T}}(k)]$. It was shown in [20] that the separating matrix $\boldsymbol{W}$ can be found by extracting the generalized eigenvectors of matrix pencil $(\boldsymbol{R}_{z}, \boldsymbol{R}_{\boldsymbol{x}})$. Hence, the BSS problem can be formulated as finding the generalized eigenvectors associated with the two sample sequences $\boldsymbol{x}(k)$ and $\boldsymbol{z}(k)$. Therefore, we can directly apply our algorithm to solve the BSS problem.

In the simulation, four benchmark signals are extracted from the file ABio7.mat provided by ICALAB [23], as shown in Fig. 8.13. We use the mixing matrix

$$\mathbf{A} = \begin{bmatrix} 2.7914 & -0.1780 & -0.4945 & 0.3013 \\ 1.3225 & -1.7841 & -0.3669 & 0.4460 \\ 0.0714 & -1.9163 & 0.4802 & -0.3701 \\ -1.7396 & 0.1302 & 0.9249 & -0.4007 \end{bmatrix}, \tag{8.178}$$

which was randomly generated. $\boldsymbol{e}[n]$ is a zero-mean white noise vector with covariance $10^{-5}\boldsymbol{I}$. Figure 8.14 shows the mixed signals. We use a simple FIR filter with coefficients $\tau = [1, -1]^{\mathrm{T}}$.

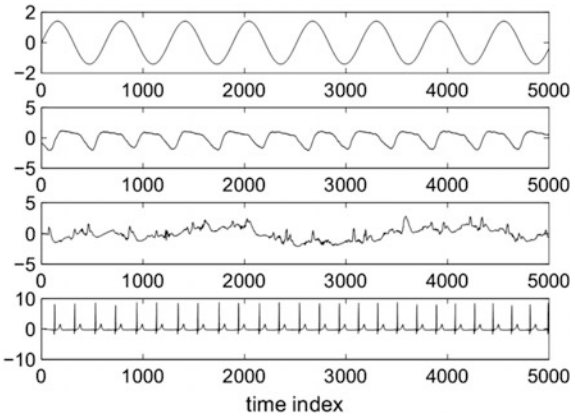

Fig. 8.13 Four original signals

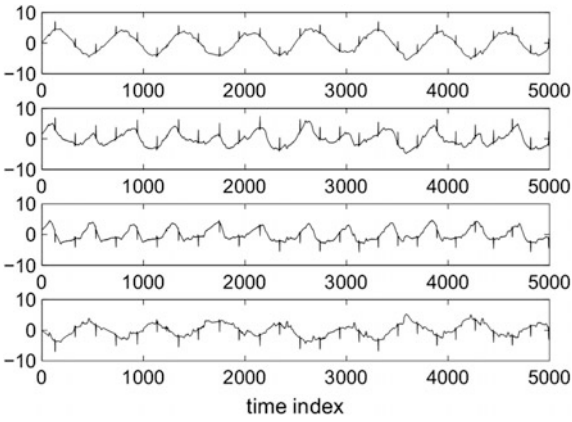

Fig. 8.14 Mixed signals

Suppose that the matrix pencil ($\boldsymbol{R}_z$, $\boldsymbol{R}_x$) has four eigenvectors $\boldsymbol{w}_1$, $\boldsymbol{w}_2$, $\boldsymbol{w}_3$, $\boldsymbol{w}_4$ associated with four eigenvalues $\sigma_1 < \sigma_2 < \sigma_3 < \sigma_4$. Thus, $\boldsymbol{B} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3, \boldsymbol{w}_4]$. We use fGPCA, nGPCA, and all other algorithms to extract the two principal generalized eigenvectors ($\boldsymbol{w}_3$ and $\boldsymbol{w}_4$). To extract the two minor generalized eigenvectors ($\boldsymbol{w}_1$ and $\boldsymbol{w}_2$), we use fGMCA, nGMCA, and gradient-based algorithms to extract the minor generalized eigenvectors of matrix pencil ($\boldsymbol{R}_z$, $\boldsymbol{R}_x$) and other algorithms to extract the principal generalized eigenvectors of matrix pencil ($\boldsymbol{R}_x$, $\boldsymbol{R}_z$). All parameters and initial values are the same as in Example 1.

Similar to Example 1, a total of $L = 100$ independent runs are evaluated in this example. The separating matrix $\boldsymbol{B}$ is calculated as $\boldsymbol{B} = (1/L)\sum_{j=1}^{L} \boldsymbol{B}_j$, where $\boldsymbol{B}_j$ is the separating matrix extracted from the jth independent run ($j = 1, 2, \ldots, L$).

Figures 8.15 to 8.16 show the recovered signals by EVD and our method, respectively. Signals separated by other algorithms are similar to Figs. 8.15 and 8.16, which are not shown in these two figures. Table 8.2 shows the absolute values

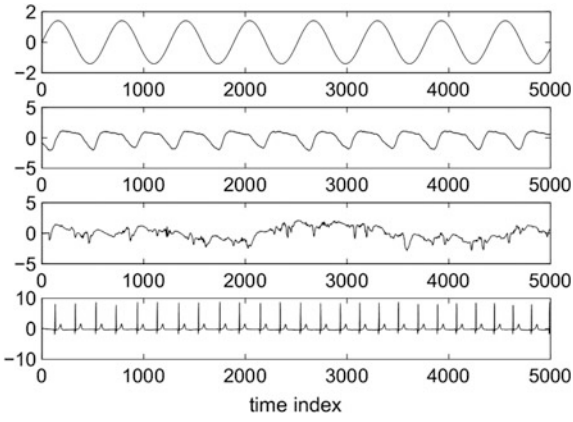

Fig. 8.15 Signals separated by EVD method

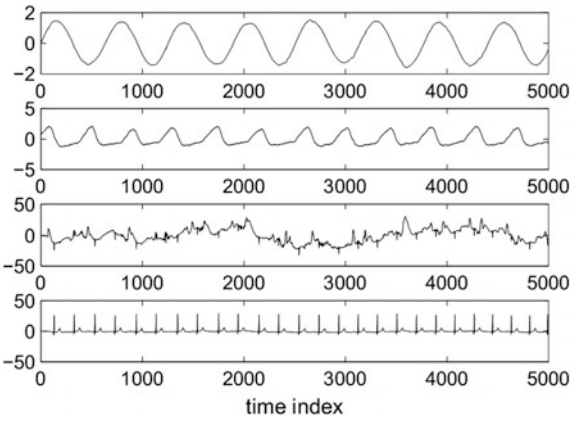

Fig. 8.16 Signals separated by proposed method

Table 8.2 Absolute values of correlation coefficients between sources and recovered signals

Method	Source 1	Source 2	Source 3	Source 4
EVD	1.0000	0.9998	0.9997	0.9989
fGM(P)CA	1.0000	0.9997	0.9992	0.9987
nGM(P)CA	1.0000	0.9996	0.9994	0.9987
Gradient-based	0.9983	0.9811	0.9989	0.9983
Power method	0.9998	0.9995	0.9991	0.9980
R-GEVE	0.9999	0.9995	0.9993	0.9988

of correlation coefficients between the sources and the recovered signals. The simulation results demonstrate that all methods can solve the BSS problem effectively, and our algorithms and the algorithms proposed in [4] can separate the signals more accurately than other algorithms. Moreover, the advantage of neural network model-based algorithms over EVD method for the BSS problem is that they are recursive algorithms and therefore can be implemented online, whereas EVD is a batch-processing method and therefore needs intensive computation.

In this section, we have derived a coupled dynamic system for GHEP based on a novel generalized information criterion. Compared with the existing work, the proposed approach is easier to obtain for that it does not need to calculate the inverse of the Hessian. Based on the dynamic system, a coupled GMCA algorithm (fGMCA) and a coupled GPCA algorithm (fGPCA) have been obtained. The convergence speed of fGMCA and fGPCA is similar to that of Nguyen's well-performed algorithms (nGMCA and nGPCA), but the computational complexity is less than that of Nguyen. Experiment results show that our algorithms have better numerical stability and can extract the generalized eigenvectors more accurately than the other algorithms.

8.5 Summary

In this chapter, the speed stability problem that plagues most noncoupled learning algorithms has been discussed and the coupled learning algorithms that are a solution for the speed stability problem have been analyzed. Moller's coupled PCA algorithm, Nguyen's coupled generalized eigen pair extraction algorithm, coupled singular value decomposition of a cross-covariance matrix, etc., have been reviewed. Then, unified and coupled algorithms for minor and principal eigen pair extraction proposed by us have been introduced, and their convergence has been analyzed. Finally, a fast and adaptive coupled generalized eigen pair extraction algorithm proposed by us has been analyzed in detail, and their convergence analysis has been proved via the DDT method.

References

1. Chen, L. H., & Chang, S. (1995). An adaptive learning algorithm for principal component analysis. *IEEE Transactions on Neural Networks, 6*(5), 1255–1263.
2. Moller, R., & Konies, A. (2004). Coupled principal component analysis. *IEEE Transactions on Neural Networks, 15*(1), 214–222.
3. Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks, 2*(6), 459–473.
4. Nguyen, T. D., & Yamada, I. (2013). Adaptive normalized quasi-Newton algorithms for extraction of generalized eigen-pairs and their convergence analysis. *IEEE Transactions on Signal Processing, 61*(6), 1404–1418.
5. Hou, L., & Chen, T. P. (2006). Online algorithm of coupled principal (minor) component analysis. *Journal of Fudan University, 45*(2), 158–169.
6. Feng, X. W., Kong, X. Y., Ma, H. G., & Liu, H. M. (2016). Unified and coupled self-stabilizing algorithm for minor and principal eigen-pair extraction. *Neural Processing Letter.* doi:10.1007/s11063-016-9520-3
7. Diamantaras, K. I., & Kung, S. Y. (1996). *Principal component neural networks. Theory and applications*. New York: Wiley.
8. Nguyen, T. D., Takahashi, N., & Yamada, I. (2013). An adaptive extraction of generalized eigensubspace by using exact nested orthogonal complement structure. *Multidimensional Systems and Signal Processing, 24*(3), 457–483.
9. Kaiser, A., Schenck, M., & Moller, R. (2010). Coupled singular value decomposition of a cross-covariance matrix. *International Journal of Neural Systems, 20*(4), 293–318.
10. Hyvarinen, A., Karhunen, J., & Oja, E. (2004). *Independent component analysis*. New Jersey: Wiley.
11. Miao, Y., & Hua, Y. (1998). Fast subspace tracking and neural network learning by a novel information criterion. *IEEE Transactions on Signal Processing, 46*(7), 1967–1979.
12. Ouyang, S., Ching, P., & Lee, T. (2002). Quasi-Newton algorithm for adaptive minor component extraction. *Electronics Letters, 38*(19), 1142–1144.
13. Golub, G. H., & Van Loan, C. F. (2012). *Matrix computations*. Baltimore: JHU Press.
14. Kong, X., Hu, C., & Han, C. (2012). A dual purpose principal and minor subspace gradient flow. *IEEE Transactions on Signal Processing, 60*(1), 197–210.
15. Peng, D., Yi, Z., & Xiang, Y. (2009). A unified learning algorithm to extract principal and minor components. *Digital Signal Processing, 19*(4), 640–649.

16. Ouyang, S., Ching, P., & Lee, T. (2003). Robust adaptive quasi-Newton algorithms for eigensubspace estimation. *IEE Proceedings-Vision, Image and Signal Processing*, *150*(5), 321–330.
17. Higham, D. J., & Higham, N. J. (2005). *MATLAB Guide*. SIAM.
18. Kong, X. Y., Hu, C. H., & Han, C. Z. (2010). On the discrete-time dynamics of a class of self-stabilizing MCA extraction algorithms. *IEEE Transactions on Neural Networks, 21*(1), 175–181.
19. Horn, R. A., & Johnson, C. R. (2012). *Matrix analysis*. Cambridge: Cambridge University Press.
20. Tom, A. M. (2006). The generalized eigendecomposition approach to the blind source separation problem. *Digital Signal Processing, 16*(3), 288–302.
21. Attallah, S., & Abed-Meraim, K. (2008). A fast adaptive algorithm for the generalized symmetric eigenvalue problem. *IEEE Signal Processing Letters, 15*, 797–800.
22. Yang, J., Chen, X., & Xi, H. (2013). Fast adaptive extraction algorithm for multiple principal generalized eigenvectors. *International Journal of Intelligent Systems, 28*(3), 289–306.
23. Icalab. Available: http://www.bsp.brain.riken.go.jp/ICALAB/
24. Tanaka, T. (2009). Fast generalized eigenvector tracking based on the power method. *IEEE Signal Processing Letters, 16*(11), 969–972.
25. Feng, X. W., Kong, X. Y., Duan, Z. S., & Ma, H. G. (2016). Adaptive generalized eigen-pairs extraction algorithms and their convergence analysis. *IEEE Transactions on Signal Processing, 64*(11), 2976–2989.