Algorithms for Accelerated Convergence of Adaptive PCA

Chanchal Chatterjee, Zhengjiu Kang, *Member, IEEE*, and Vwani P. Roychowdhury

***Abstract*—We derive and discuss new adaptive algorithms for principal component analysis (PCA) that are shown to converge faster than the traditional PCA algorithms due to Oja, Sanger, and Xu. It is well known that traditional PCA algorithms that are derived by using gradient descent on an objective function are slow to converge. Furthermore, the convergence of these algorithms depends on appropriate choices of the gain sequences. Since online applications demand faster convergence and an automatic selection of gains, we present new adaptive algorithms to solve these problems. We first present an unconstrained objective function, which can be minimized to obtain the principal components. We derive adaptive algorithms from this objective function by using: 1) gradient descent; 2) steepest descent; 3) conjugate direction; and 4) Newton–Raphson methods. Although gradient descent produces Xu's LMSER algorithm, the steepest descent, conjugate direction, and Newton–Raphson methods produce new adaptive algorithms for PCA. We also provide a discussion on the landscape of the objective function, and present a global convergence proof of the adaptive gradient descent PCA algorithm using stochastic approximation theory. Extensive experiments with stationary and nonstationary multidimensional Gaussian sequences show faster convergence of the new algorithms over the traditional gradient descent methods. We also compare the steepest descent adaptive algorithm with state-of-the-art methods on stationary and nonstationary sequences.**

***Index Terms*—Adaptive principal component analysis, eigendecomposition, principal subspace analysis.**

I. Introduction

RECENT research in the area of adaptive principal component analysis (PCA) has produced a large number of algorithms that are derived from: 1) anti-Hebbian learning [2], [12]; 2) Hebbian learning [2], [12]; 3) lateral interaction algorithms [2], [12]; and 4) gradient-based learning [2], [3], [6]–[10], [12], [15]–[18], [20]–[25]. Such algorithms are used in a variety of applications in signal and image processing. For example, in digital image processing, PCA algorithms are used for data compression, feature extraction and dimensionality reduction in neural-network training. In signal processing, PCA algorithms are applied successfully to both temporal and spatial domain spectral analyzes. Examples include multiple signal classification (MUSIC) techniques, minimum-norm methods, ESPIRIT estimator, and weighted subspace fitting (WSF) methods for estimating frequencies of sinusoids or direction of arrival (DOA) of plane waves impinging on an antenna array.

Although most PCA implementations are based on the gradient descent technique on an objective function, it is well known [2], [5], [12] that such traditional PCA algorithms are slow to converge. Furthermore, both analytical and experimental studies show that convergence of these algorithms depend on appropriate selection of the gain sequence $\{\eta_k\}$ (see Section I-A). Moreover, it is proven [5], [6], [17] that if the gain sequence exceeds an upper bound, then the algorithms may diverge or converge to a false solution.

Since most of these algorithms are used for real-time (i.e., online) processing, it is especially difficult to determine an appropriate choice of the gain parameter at the start of the online process. Hence, it is important, for wider applicability of these algorithms, to 1) speed up the convergence of the algorithms and 2) automatically select the gain parameter based on the current data sample.

A. Objective Functions for Gradient-Based Adaptive PCA

In this section, we present a number of common objective functions from which we can derive various adaptive PCA algorithms by using nonlinear optimization techniques such as: 1) gradient descent; 2) steepest descent; 3) conjugate direction; 4) Newton–Raphson (NR); and 5) recursive least squares (RLS). Note that, each optimization method when applied to a different objective function, leads to a new algorithm for adaptive PCA.

The first objective function J_1 is used extensively in signal processing applications and is derived from the Rayleigh quotient as follows:

$$J_1(\boldsymbol{w}) = -\frac{\boldsymbol{w}^T A \boldsymbol{w}}{\boldsymbol{w}^T \boldsymbol{w}} \tag{1}$$

where A is a symmetric, positive definite matrix whose principal eigenvectors need to be computed. Sarkar *et al.* [21], [24] applied the steepest descent and conjugate direction methods on J_1 to compute the extremal (largest or smallest) eigenvectors of A. The algorithms can be easily modified for a stationary or nonstationary sequence $\{A_k\}$ instead of a constant matrix A. Fu and Dowling [9] generalized the conjugate direction algorithm to compute all eigenvectors of A. Zhu and Wang [25] also used a conjugate direction method on a regularized total least squares version of this objective function. A survey of conjugate direction-based algorithms on J_1 can be found in [24].

Objective function J_1 can be alternatively written as the following Lagrangian:

$$\tilde{J}_1(\boldsymbol{w}) = -\boldsymbol{w}^T A \boldsymbol{w} + \alpha(\boldsymbol{w}^T \boldsymbol{w} - 1) \tag{2}$$

Manuscript received January 29, 1999; revised October 14, 1999.

C. Chatterjee is with BAE Systems Inc., San Diego, CA 92127 USA (e-mail: Cchatterj@hotmail.com).

Z. Kang and V. P. Roychowdhury are with the Electrical Engineering Department, University of California at Los Angeles, Los Angeles, CA 90095 USA (e-mail: vwani@ee.ucla.edu).

Publisher Item Identifier S 1045-9227(00)02996-9.

where α is a Lagrange multiplier. Equating the gradient of $\tilde{J}_1$ (with respect to $\boldsymbol{w}$) to zero, and applying the constraint $\boldsymbol{w}^T\boldsymbol{w} = 1$, we obtain $\alpha = \boldsymbol{w}^T A\boldsymbol{w}$, from which we obtain the following adaptive gradient descent PCA algorithm:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \eta_k(A_k\boldsymbol{w}_k - \boldsymbol{w}_k^T A_k\boldsymbol{w}_k\boldsymbol{w}_k) \tag{3}$$

where $\{\eta_k\}$ is a sequence of scalar gains, and A_k is a online observation for A. Note that this is the well-known Oja's algorithm [17] for the evaluation of the first principal eigenvector of A.

The second objective function J_2 is obtained by using the penalty function method on $\tilde{J}_1$ as follows:

$$J_2(\boldsymbol{w}) = -\boldsymbol{w}^T A\boldsymbol{w} + \alpha(\boldsymbol{w}^T\boldsymbol{w} - 1)^2 \tag{4}$$

where α is a nonnegative scalar constant. Chauvin [7] presents a gradient descent algorithm based on J_2, and analyzes the landscape of this function. Mathew *et al.* [15] also use this objective function to offer a Newton-type algorithm for adaptive PCA.

The third objective function J_3 is based on an information-theoretic criterion whose landscape has been studied by Plumbley [18] and Miao and Hua [16]

$$J_3(\boldsymbol{w}) = \log(\boldsymbol{w}^T A\boldsymbol{w}) - \boldsymbol{w}^T\boldsymbol{w}. \tag{5}$$

Plumbley analyzes a variation of J_3 and presents a gradient descent algorithm. Miao and Hua present gradient descent and RLS algorithms for adaptive principal subspace analysis (PSA).

The last objective function J_4 is the focus of our study. It has been studied in different forms in the literature. For example, Xu [22], Yang [23], Fu and Dowling [10], Bannour and Azimi-Sadjadi [3], and Miao and Hua [16] use the following objective function:

$$J_4(\boldsymbol{w}) = \sum_{t=1}^{k} \beta^{k-t}|\boldsymbol{x}_t - \boldsymbol{w}\boldsymbol{w}^T\boldsymbol{x}_t|^2 \tag{6}$$

where $\boldsymbol{x}_t$ is an observation vector at time t, and $0 < \beta \leq 1$ is a forgetting factor used for nonstationary observation sequences. As discussed in Section II, this objective function is similar to the mean squared error at the output of a two-layer linear autoassociative neural network. Xu [22] derives an algorithm for adaptive PSA by using gradient descent on J_4. Yang [23] proposes gradient descent and RLS algorithms using J_4. Bannour and Azimi-Sadjadi [3] also describe an RLS algorithm for adaptive PCA with J_4. Fu and Dowling [10] reduce J_4 to an objective function similar to J_1, which can be minimized by the conjugate direction methods due to Sarkar *et al.* [21], [24]. They also compute the minor components by using an approximation $\boldsymbol{x}_t \approx \boldsymbol{w}\boldsymbol{w}^T\boldsymbol{x}_t$, and by employing the deflation technique.

B. Outline of the Study

In this study, we present an unconstrained objective function for adaptive PCA and show that it is similar to J_4. We then derive various adaptive algorithms for PCA by using the 1) gradient descent; 2) steepest descent; 3) conjugate direction; and 4) NR methods of nonlinear optimization on this objective function. For RLS-type adaptive PCA algorithms, we refer to [3], [16], and [23]. Although gradient descent produces the well-known Xu's least mean square error reconstruction (LMSER) algorithm [22], the steepest descent, conjugate direction, and NR methods produce new adaptive algorithms for PCA. We also discuss the landscape of the objective function, and present a global convergence proof of the adaptive gradient descent PCA algorithm by using stochastic approximation theory.

Finally, we apply these algorithms on stationary and nonstationary multidimensional Gaussian data sequences. The correlation matrix A is presented to the algorithms as an online process $\{A_k\}$. We experimentally show that the adaptive steepest descent, conjugate direction and NR algorithms converge much faster than the traditional gradient descent technique due to Xu [22]. Furthermore, the new algorithms automatically select the gain sequence $\{\eta_k\}$ based on the current data sample. We further compare the steepest descent algorithm with state-of-the-art methods such as Yang's projection approximation subspace tracking (PASTd) [23], Bannour and Sadjadi's RLS [3], and Fu and Dowling's conjugate gradient eigenstructure tracking (CGET1) [9], [10] algorithms.

In Section II, we present an unconstrained objective function for PCA. We also demonstrate that this objective function is similar to the mean squared error at the output of a two-layer linear autoassociative network. In Section III, we apply the gradient descent, steepest descent, conjugate direction, and NR optimization methods to this objective function to obtain new algorithms for adaptive PCA. In Section IV, we provide an analysis of the landscape of the objective function, and a convergence proof of the adaptive gradient descent algorithm. In Section V, we present experimental results with stationary and nonstationary Gaussian sequences, thereby showing faster convergence of the new algorithms over the traditional gradient descent adaptive PCA algorithm. We also compare the steepest descent algorithm with state-of-the-art algorithms.

II. Derivation of Objective Function

It is well known [2] that the standard quadratic problem leading to a PCA solution is one of how to minimize the objective function

$$-\boldsymbol{w}_i^T A\boldsymbol{w}_i \text{ under constraints } \boldsymbol{w}_i^T\boldsymbol{w}_j = \delta_{ij} \qquad \text{for } j = 1, \cdots, i \tag{7}$$

where the optimal value of $\boldsymbol{w}_i \in \Re^n$ is the ith principal eigenvector of a symmetric, positive definite matrix $A \in \Re^{n\times n}$, and δ_{ij} is the Kronecker's delta. (7) is equivalent to the unconstrained minimization problem

$$\tilde{J}(\boldsymbol{w}_i;\, A,\, \alpha,\, \beta) = -\boldsymbol{w}_i^T A\boldsymbol{w}_i + \alpha(\boldsymbol{w}_i^T\boldsymbol{w}_i - 1) + 2\sum_{j=1}^{i-1} \beta_j \boldsymbol{w}_i^T\boldsymbol{w}_j \tag{8}$$

where α and β_j $(j = 1, \cdots, i-1)$ are the Lagrange multipliers. By equating the gradient of $\tilde{J}$ with respect to $\boldsymbol{w}_i$ to zero, we obtain

$$(1/2)\nabla_{\boldsymbol{w}_i}\tilde{J}(\boldsymbol{w}_i;\, A,\, \alpha,\, \beta) = -A\boldsymbol{w}_i + \alpha\boldsymbol{w}_i + \sum_{j=1}^{i-1} \beta_j \boldsymbol{w}_j = 0. \tag{9}$$

By multiplying $\nabla_{\boldsymbol{w}_i}\tilde{J}(\boldsymbol{w}_i)$ to the left by $\boldsymbol{w}_i^T$ and applying the constraints, we obtain

$$\alpha = \boldsymbol{w}_i^T A \boldsymbol{w}_i. \tag{10}$$

Similarly, by multiplying $\nabla_{\boldsymbol{w}_i}\tilde{J}(\boldsymbol{w}_i)$ to the left by $\boldsymbol{w}_j^T$ and applying the constraints, we obtain

$$\beta_j = \boldsymbol{w}_j^T A \boldsymbol{w}_i \text{ for } j = 1, \cdots, i-1. \tag{11}$$

Replacing these values of the Lagrange multipliers into (8), we obtain a new unconstrained objective function

$$J(\boldsymbol{w}_i;\, A) = -2\boldsymbol{w}_i^T A \boldsymbol{w}_i + \boldsymbol{w}_i^T A \boldsymbol{w}_i \boldsymbol{w}_i^T \boldsymbol{w}_i + 2\sum_{j=1}^{i-1} \boldsymbol{w}_i^T \boldsymbol{w}_j \boldsymbol{w}_j^T A \boldsymbol{w}_i. \tag{12}$$

An online version of this objective function can be obtained as

$$J(\boldsymbol{w}_k^i;\, A_k) = -2\boldsymbol{w}_k^{i^T} A_k \boldsymbol{w}_k^i + \boldsymbol{w}_k^{i^T} A_k \boldsymbol{w}_k^i \boldsymbol{w}_k^{i^T} \boldsymbol{w}_k^i + 2\sum_{j=1}^{i-1} \boldsymbol{w}_k^{i^T} \boldsymbol{w}_k^j \boldsymbol{w}_k^{j^T} A_k \boldsymbol{w}_k^i \tag{13}$$

where A_k is the online observation for A.

A. Similarities with Other Objective Functions

The objective function (13) is similar to several other objective functions in signal processing and neural-networks literature. For example, Yang [23], Fu and Dowling [10], Bannour and Azimi-Sadjadi [3], and Miao and Hua [16] use the following objective function:

$$J_4(\boldsymbol{w}) = \sum_{t=1}^{k} \beta^{k-t} |\boldsymbol{x}_t - \boldsymbol{w}\boldsymbol{w}^T \boldsymbol{x}_t|^2 \tag{14}$$

where $\boldsymbol{x}_t \in \Re^n$ is an observation vector at time t, and $0 < \beta \leq 1$ is a forgetting factor intended to ensure that data in the distant past are down-weighted in order to afford the tracking capability when the algorithm operates with nonstationary data sequences.

We next show that the objective function J_4 is same as the output sum squared error of a two-layer linear autoassociative neural network. Autoassociation is a neural-network structure in which the desired output is same as the network input $\boldsymbol{x}_t$. For a two-layer linear network with p nodes in the hidden layer, if we denote the optimal weight matrices of the input and output layers by W_1 and W_2, respectively, then we can show [12]

$$W_1 = R\Phi_p \quad \text{and} \quad W_2 = \Phi_p^T R^{-1} \tag{15}$$

where Φ_p is a matrix whose rows are the p principal eigenvectors of the input correlation matrix $A_k = (1/k)\sum_{t=1}^{k} \beta^{k-t} \boldsymbol{x}_t \boldsymbol{x}_t^T$, and R is a nonsingular $n \times n$ matrix. Here, "optimality" refers to the minimum mean squared error at the network output. If we further impose the constraint $W_2 = W_1^T$, then R is a unitary matrix and the input layer weight matrix W_1 is orthonormal and spans the space defined by the p principal eigenvectors of the input correlation matrix A_k.

Note that if we have a single node in the hidden layer (i.e., $p = 1$), then we obtain J_4 as the output sum squared error for a two-layer linear autoassociative network with input layer weight vector $\boldsymbol{w}$ and output layer weight vector $\boldsymbol{w}^T$. Furthermore, the optimal value of $\boldsymbol{w}$ is the first principal eigenvector of the input correlation matrix A_k. Instead, of a single neuron in the hidden layer, if we have $p\,(>1)$ hidden neurons, then we need to modify the objective function J_4 such that the optimal value of the input layer weight matrix W_1 is Φ_p. In other words, if $\boldsymbol{w}_i$ is the ith row vector of W_1 then the optimal value of $\boldsymbol{w}_i$ is the ith principal eigenvector of A_k. For this, we modify the desired output corresponding to ith hidden neuron as

$$\boldsymbol{d}_t^i = \boldsymbol{x}_t - \sum_{j=1}^{i-1} \boldsymbol{w}_j \boldsymbol{w}_j^T \boldsymbol{x}_t \tag{16}$$

where $\boldsymbol{w}_j$ is the input weight vector for the jth ($j < i$) neuron. This process is same as the "deflation" [10]–[12] of the desired output instead of the original input $\boldsymbol{x}_t$ in the autoassociative model. The resultant objective function is

$$J_4(\boldsymbol{w}_k^i) = \sum_{t=1}^{k} \beta^{k-t} \left\| x_t - \sum_{j=1}^{i-1} \boldsymbol{w}_k^j \boldsymbol{w}_k^{j^T} \boldsymbol{x}_t - \boldsymbol{w}_k^i \boldsymbol{w}_k^{i^T} \boldsymbol{x}_t \right\|^2. \tag{17}$$

Simplifying this equation, we obtain

$$J_4(\boldsymbol{w}_k^i) = \sum_{t=1}^{k} \beta^{k-t} \left\| \boldsymbol{x}_t - \sum_{j=1}^{i-1} \boldsymbol{w}_k^j \boldsymbol{w}_k^{j^T} \boldsymbol{x}_t \right\|^2 - 2\boldsymbol{w}_k^{i^T} A_k \boldsymbol{w}_k^i + \boldsymbol{w}_k^{i^T} A_k \boldsymbol{w}_k^i \boldsymbol{w}_k^{i^T} \boldsymbol{w}_k^i + 2\sum_{j=1}^{i-1} \boldsymbol{w}_k^{j^T} A_k \boldsymbol{w}_k^i \boldsymbol{w}_k^{i^T} \boldsymbol{w}_k^j$$

which is same as the objective function $J(\boldsymbol{w}_k^i;\, A_k)$ in (13).

Another common technique of computing minor components from an objective function is to use deflation of the correlation matrix A. We know from (2), that the objective function for computing the first principal component of A is

$$\tilde{J}(\boldsymbol{w}_1;\, A,\, \alpha) = -\boldsymbol{w}_1^T A \boldsymbol{w}_1 + \alpha(\boldsymbol{w}_1^T \boldsymbol{w}_1 - 1).$$

In order to compute the minor components $\boldsymbol{w}_i$ ($i > 1$), we use the following objective function:

$$\tilde{J}(\boldsymbol{w}_i;\, \tilde{A}_i,\, \alpha) = -\boldsymbol{w}_i^T \tilde{A}_i \boldsymbol{w}_i + \alpha(\boldsymbol{w}_i^T \boldsymbol{w}_i - 1) \tag{18}$$

where $\tilde{A}_i$ is a deflation of A as follows:

$$\tilde{A}_i = A - \xi \sum_{j=1}^{i-1} \boldsymbol{w}_k^j \boldsymbol{w}_k^{j^T} A \tag{19}$$

for $\xi \geq 1$. By using $\nabla_{\boldsymbol{w}_i}\tilde{J}(\boldsymbol{w}_i;\, \tilde{A}_i,\, \alpha) = 0$ and applying the constraints $\boldsymbol{w}_i^T \boldsymbol{w}_j = \delta_{ij}$ for $j = 1, \cdots, i$, we obtain $\alpha = \boldsymbol{w}_i^T A \boldsymbol{w}_i$. By replacing this value of α into $\tilde{J}(\boldsymbol{w}_i;\, \tilde{A}_i,\, \alpha)$, we obtain

$$\tilde{J}(\boldsymbol{w}_i;\, A) = -2\boldsymbol{w}_i^T A \boldsymbol{w}_i + \boldsymbol{w}_i^T A \boldsymbol{w}_i \boldsymbol{w}_i^T \boldsymbol{w}_i + \xi \sum_{j=1}^{i-1} \boldsymbol{w}_i^T \boldsymbol{w}_j \boldsymbol{w}_j^T A \boldsymbol{w}_i$$

which is same as $J(\boldsymbol{w}_i;\, A)$ in (12) for $\xi = 2$.

III. New Algorithms for Adaptive PCA

We now apply different methods of nonlinear minimization to the objective function $J(\boldsymbol{w}_k^i;\, A_k)$ in (13) to obtain various algorithms for adaptive PCA.

A. Gradient Descent Method

We first apply the gradient descent method on the objective function $J(\boldsymbol{w}_k^i;\, A_k)$ to obtain

$$\begin{aligned} \boldsymbol{g}_k^i &= (1/2)\nabla_{\boldsymbol{w}_i} J(\boldsymbol{w}_k^i;\, A_k) \\ &= -2A_k\boldsymbol{w}_k^i + \boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}A_k\boldsymbol{w}_k^i + A_k\boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}\boldsymbol{w}_k^i \\ &\quad + \sum_{j=1}^{i-1}\boldsymbol{w}_k^j\boldsymbol{w}_k^{j^T}A_k\boldsymbol{w}_k^i + \sum_{j=1}^{i-1}A_k\boldsymbol{w}_k^j\boldsymbol{w}_k^{j^T}\boldsymbol{w}_k^i. \end{aligned} \tag{20}$$

This gives us the following adaptive gradient descent algorithm for PCA

$$\boldsymbol{w}_{k+1}^i = \boldsymbol{w}_k^i - \eta_k\boldsymbol{g}_k^i \tag{21}$$

or

$$\begin{aligned} \boldsymbol{w}_{k+1}^i = \boldsymbol{w}_k^i + \eta_k \Bigg(& 2A_k\boldsymbol{w}_k^i - \boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}A_k\boldsymbol{w}_k^i \\ & - A_k\boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}\boldsymbol{w}_k^i - \sum_{j=1}^{i-1}\boldsymbol{w}_k^j\boldsymbol{w}_k^{j^T}A_k\boldsymbol{w}_k^i \\ & - \sum_{j=1}^{i-1}A_k\boldsymbol{w}_k^j\boldsymbol{w}_k^{j^T}\boldsymbol{w}_k^i \Bigg) \end{aligned}$$

where $\{\eta_k\}$ is a sequence of scalar gains whose properties are defined in Section IV-B. It is convenient to define a matrix $W_k = [\boldsymbol{w}_k^1 \ \cdots \ \boldsymbol{w}_k^p]$ ($p \le n$), for which the columns are the p weight vectors that converge to the p principal eigenvectors of A_k, respectively. Then, (21) can be represented as

$$\begin{aligned} W_{k+1} = W_k + \eta_k \big(& 2A_kW_k - W_k\mathrm{UT}\left(W_k^TA_kW_k\right) \\ & - A_kW_k\mathrm{UT}\left(W_k^TW_k\right)\big) \end{aligned} \tag{22}$$

where UT[·] sets all elements below the diagonal of its matrix argument to zero, thereby making it upper triangular. Note that (21) is the LMSER algorithm due to Xu [22] that was derived from a least mean squared error criterion of a feedforward neural network.

B. Steepest Descent Method

The adaptive steepest descent algorithm for PCA is obtained from $J(\boldsymbol{w}_k^i;\, A_k)$ in (13) as

$$\boldsymbol{w}_{k+1}^i = \boldsymbol{w}_k^i - \alpha_k^i\boldsymbol{g}_k^i \tag{23}$$

where $\boldsymbol{g}_k^i$ is given in (20), and α_k^i is a nonnegative scalar minimizing $J(\boldsymbol{w}_k^i - \alpha\boldsymbol{g}_k^i;\, A_k)$. Since we have an expression for $J(\boldsymbol{w}_k^i;\, A_k)$ in (13), we minimize the function $J(\boldsymbol{w}_k^i - \alpha\boldsymbol{g}_k^i;\, A_k)$ with respect to α and obtain the following cubic equation:

$$c_3\alpha^3 + c_2\alpha^2 + c_1\alpha + c_0 = 0 \tag{24}$$

where

$$\begin{aligned} c_0 &= -\boldsymbol{g}_k^{i^T}\boldsymbol{g}_k^i; \\ c_1 &= \boldsymbol{g}_k^{i^T}H_k^i\boldsymbol{g}_k^i; \\ c_2 &= -3\left(\boldsymbol{g}_k^{i^T}A_k\boldsymbol{g}_k^i\boldsymbol{w}_k^{i^T}\boldsymbol{g}_k^i + \boldsymbol{w}_k^{i^T}A_k\boldsymbol{g}_k^i\boldsymbol{g}_k^{i^T}\boldsymbol{g}_k^i\right); \\ c_3 &= 2\boldsymbol{g}_k^{i^T}A_k\boldsymbol{g}_k^i\boldsymbol{g}_k^{i^T}\boldsymbol{g}_k^i. \end{aligned}$$

Here H_k^i is the Hessian of $J(\boldsymbol{w}_k^i;\, A_k)$ as given below

$$\begin{aligned} H_k^i = & -2A_k + 2A_k\boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T} + 2\boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}A_k \\ & + A_k\boldsymbol{w}_k^{i^T}\boldsymbol{w}_k^i + \boldsymbol{w}_k^{i^T}A_k\boldsymbol{w}_k^iI \\ & + \sum_{j=1}^{i-1}A_k\boldsymbol{w}_k^j\boldsymbol{w}_k^{j^T} + \sum_{j=1}^{i-1}\boldsymbol{w}_k^j\boldsymbol{w}_k^{j^T}A_k. \end{aligned} \tag{25}$$

With known values of $\boldsymbol{w}_k^i$ and $\boldsymbol{g}_k^i$, the above cubic equation can be solved to obtain α that minimizes $J(\boldsymbol{w}_k^i - \alpha\boldsymbol{g}_k^i;\, A_k)$. A description of the computation and choice of α is given in the Appendix.

We now represent the adaptive PCA algorithm (23) in the matrix form. We define the matrices

$$\begin{aligned} W_k &= [\boldsymbol{w}_k^1, \cdots, \boldsymbol{w}_k^p] \\ G_k &= [\boldsymbol{g}_k^1, \cdots, \boldsymbol{g}_k^p] \end{aligned}$$

and

$$\Gamma_k = \mathrm{diag}(\alpha_k^1, \cdots, \alpha_k^p).$$

Then, the adaptive steepest descent PCA algorithm is

$$\begin{aligned} G_k &= -2A_kW_k + W_k\mathrm{UT}\left(W_k^TA_kW_k\right) \\ &\quad + A_kW_k\mathrm{UT}\left(W_k^TW_k\right) \\ W_{k+1} &= W_k - G_k\Gamma_k. \end{aligned} \tag{26}$$

Here UT[·] is same as in (22).

C. Conjugate Direction Method

The adaptive conjugate direction algorithm for PCA can be obtained as follows:

$$\begin{aligned} \boldsymbol{w}_{k+1}^i &= \boldsymbol{w}_k^i + \alpha_k^i\boldsymbol{d}_k^i \\ \boldsymbol{d}_{k+1}^i &= -\boldsymbol{g}_{k+1}^i + \beta_k^i\boldsymbol{d}_k^i \end{aligned} \tag{27}$$

where $\boldsymbol{g}_{k+1}^i = (1/2)\nabla_{\boldsymbol{w}_i}J(\boldsymbol{w}_{k+1}^i;\, A_k)$. The gain constant α_k^i is chosen as α that minimizes $J(\boldsymbol{w}_k^i + \alpha\boldsymbol{d}_k^i)$. Similar to the steepest descent case, we obtain the following cubic equation:

$$c_3\alpha^3 + c_2\alpha^2 + c_1\alpha + c_0 = 0 \tag{28}$$

where

$$\begin{aligned} c_0 &= \boldsymbol{g}_k^{i^T}\boldsymbol{d}_k^i; \\ c_1 &= \boldsymbol{d}_k^{i^T}H_k^i\boldsymbol{d}_k^i; \\ c_2 &= 3\left(\boldsymbol{d}_k^{i^T}A_k\boldsymbol{d}_k^i\boldsymbol{w}_k^{i^T}\boldsymbol{d}_k^i + \boldsymbol{w}_k^{i^T}A_k\boldsymbol{d}_k^i\boldsymbol{d}_k^{i^T}\boldsymbol{d}_k^i\right); \\ c_3 &= 2\boldsymbol{d}_k^{i^T}A_k\boldsymbol{d}_k^i\boldsymbol{d}_k^{i^T}\boldsymbol{d}_k^i. \end{aligned}$$

Here, $\boldsymbol{g}_k^i = (1/2)\nabla_{\boldsymbol{w}_i}J(\boldsymbol{w}_k^i;\, A_k)$ as given in (20). The above equation is solved to obtain α that minimizes $J(\boldsymbol{w}_k^i + \alpha\boldsymbol{d}_k^i)$.

For the choice of β_k^i, we can use a number of methods [14] as described below

Hestenes–Stiefel:
$$\beta_k^i = \boldsymbol{g}_{k+1}^{i^T}\left(\boldsymbol{g}_{k+1}^i - \boldsymbol{g}_k^i\right) / \boldsymbol{d}_k^{i^T}\left(\boldsymbol{g}_{k+1}^i - \boldsymbol{g}_k^i\right)$$
Polak–Ribiere:
$$\beta_k^i = \boldsymbol{g}_{k+1}^{i^T}\left(\boldsymbol{g}_{k+1}^i - \boldsymbol{g}_k^i\right) / \boldsymbol{g}_k^{i^T}\boldsymbol{g}_k^i$$
Fletcher–Reeves:
$$\beta_k^i = \boldsymbol{g}_{k+1}^{i^T}\boldsymbol{g}_{k+1}^i \Big/ \boldsymbol{g}_k^{i^T}\boldsymbol{g}_k^i$$
Powell:
$$\beta_k^i = \max\left[0, \boldsymbol{g}_{k+1}^{i^T}\left(\boldsymbol{g}_{k+1}^i - \boldsymbol{g}_k^i\right) / \boldsymbol{g}_k^{i^T}\boldsymbol{g}_k^i\right].$$

We now represent the adaptive conjugate direction PCA algorithm (27) in the matrix form. We define the following matrices:

$$W_k = [\boldsymbol{w}_k^1, \cdots, \boldsymbol{w}_k^p],\ G_k = [\boldsymbol{g}_k^1, \cdots, \boldsymbol{g}_k^p],\ D_k = [\boldsymbol{d}_k^1, \cdots, \boldsymbol{d}_k^p]$$
$$\Gamma_k = \mathrm{diag}(\alpha_k^1, \cdots, \alpha_k^p), \quad \text{and} \quad \Pi_k = \mathrm{diag}(\beta_k^1, \cdots, \beta_k^p).$$

Then, the adaptive conjugate direction PCA algorithm is

$$\begin{aligned} W_{k+1} &= W_k + D_k\Gamma_k \\ G_{k+1} &= -2A_kW_{k+1} + W_{k+1}\mathrm{UT}\left(W_{k+1}^TA_kW_{k+1}\right) \\ &\quad + A_kW_{k+1}\mathrm{UT}\left(W_{k+1}^TW_{k+1}\right) \\ D_{k+1} &= -G_{k+1} + D_k\Pi_k. \end{aligned} \tag{29}$$

Here UT[·] is same as in (22).

D. Newton–Raphson Method

The adaptive NR algorithm for PCA is

$$\boldsymbol{w}_{k+1}^i = \boldsymbol{w}_k^i - \alpha_k^i\left(H_k^i\right)^{-1}g_k^i \tag{30}$$

where α_k^i is a nonnegative scalar, and H_k^i is the online Hessian given in (25). The search parameter α_k^i is commonly selected: 1) by minimizing $J(\boldsymbol{w}_k^i + \alpha\boldsymbol{d}_k^i)$ where $\boldsymbol{d}_k^i = -(H_k^i)^{-1}\boldsymbol{g}_k^i$ as shown in Section III-C; 2) as a scalar constant; or 3) as a decreasing sequence $\{\alpha_k^i\}$ such that $\alpha_k^i \to 0$ as $k \to \infty$.

The main concerns in this algorithm are that H_k^i should be positive definite, and that we should adaptively obtain an estimate of $(H_k^i)^{-1}$ in order to make the algorithm computationally efficient. These two concerns are addressed if we approximate the Hessian by dropping the term $(-A_k + A_k\boldsymbol{w}_k^{i^T}\boldsymbol{w}_k^i)$, which is close to zero for $\boldsymbol{w}_k^i$ close to the solution. The new Hessian is

$$H_k^i \approx \boldsymbol{w}_k^{i^T}A_k\boldsymbol{w}_k^iI - \tilde{A}_k^i + 2A_k\boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T} + 2\boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}A_k \tag{31}$$

where

$$\tilde{A}_k^i = A_k - \sum_{j=1}^{i-1}\boldsymbol{w}_k^j\boldsymbol{w}_k^{j^T}A_k - \sum_{j=1}^{i-1}A_k\boldsymbol{w}_k^j\boldsymbol{w}_k^{j^T}.$$

We can compute A_k^i by an iterative equation in i as follows:

$$\tilde{A}_k^{i+1} = \tilde{A}_k^i - \boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}A_k - A_k\boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}.$$

Inverting this Hessian consists of inverting the matrix $B_k^i = \boldsymbol{w}_k^{i^T}A_k\boldsymbol{w}_k^iI - \tilde{A}_k^i$, and two rank-one updates. An approximate inverse of this matrix B_k^i is given by

$$\left(B_k^i\right)^{-1} = \left(\boldsymbol{w}_k^{i^T}A_k\boldsymbol{w}_k^iI - \tilde{A}_k^i\right)^{-1} \approx \frac{I + \tilde{A}_k^i\Big/\boldsymbol{w}_k^{i^T}A_k\boldsymbol{w}_k^i}{\boldsymbol{w}_k^{i^T}A_k\boldsymbol{w}_k^i}. \tag{32}$$

An adaptive algorithm for inverting the Hessian H_k^i in (31) can be obtained by two rank-one updates. Let us define

$$C_k^i = B_k^i + 2A_k\boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}. \tag{33}$$

Then from (31), an update formula for $(H_k^i)^{-1}$ is

$$\left(H_k^i\right)^{-1} = \left(C_k^i\right)^{-1} - \frac{2\left(C_k^i\right)^{-1}\boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}A_k\left(C_k^i\right)^{-1}}{1 + 2\boldsymbol{w}_k^{i^T}A_k\left(C_k^i\right)^{-1}\boldsymbol{w}_k^i} \tag{34}$$

where $(C_k^i)^{-1}$ is obtained from (33) as:

$$\left(C_k^i\right)^{-1} = \left(B_k^i\right)^{-1} - \frac{2\left(B_k^i\right)^{-1}A_k\boldsymbol{w}_k^i\boldsymbol{w}_k^{i^T}\left(B_k^i\right)^{-1}}{1 + 2\boldsymbol{w}_k^{i^T}\left(B_k^i\right)^{-1}A_k\boldsymbol{w}_k^i} \tag{35}$$

and $(B_k^i)^{-1}$ is given in (32).

IV. Convergence Analysis

Although the adaptive PCA algorithms are derived by using standard optimization methods on objective function (13), their derivations do not constitute a proof of convergence. Hence, it is important to provide a convergence analysis for the algorithms described above. In simple terms, our derivations of the algorithms from objective function (12) show that the descent direction of (12) is the same as the average evolution directions of the adaptive algorithms. We still need to show that the global minimum of the objective function (12) is the principal eigenvector matrix of A. In this section, we prove this result. We next, use stochastic approximation theory [4], [13] to prove the global convergence of the gradient descent adaptive algorithm (22).

A. Landscape of Objective Function (12)

Here we study the landscape of objective function $J(\boldsymbol{w}_i; A)$ in (12). We make the following assumption.

Assumption A1: Matrix A is symmetric, real,[1] and positive definite. The p ($\leq n$) largest eigenvalues of A are each of unit multiplicity.

In the following discussion, we denote $\lambda_1 > \lambda_2 > \cdots > \lambda_p > \lambda_{p+1} \geq \cdots \geq \lambda_n > 0$ as the eigenvalues of A, and ϕ_i as the eigenvector corresponding to λ_i such that $\phi_1, \cdots, \phi_n$ are orthonormal. Notice that if ϕ_i is an eigenvector of A, then $d_i\phi_i$ ($d_i = \pm 1$) is also an eigenvector of A. We use the subscript (i) to denote the ith permutation of the indexes $\{1, 2, \cdots, n\}$.

In order to satisfy the first-order conditions for the existence of the equilibrium points of the joint objective

[1]If A is a Hermitian matrix, then the complex Hermitian $n \times n$ eigendecomposition problem can be reduced to a real, symmetric $2n \times 2n$ problem [11].

functions $J(\boldsymbol{w}_i; A)$ for $i = 1, \cdots, p$, we need to find a $W^* = [\boldsymbol{w}_1^* \ \boldsymbol{w}_2^* \ \cdots \ \boldsymbol{w}_p^*]$ such that

$$\nabla_{\boldsymbol{w}_j} J(\boldsymbol{w}_i^*; A) = 0 \qquad \text{for } i, j = 1, \cdots, p.$$

Theorem 1: Let A1 hold. Then, all the equilibrium points of the joint objective functions $J(\boldsymbol{w}_i; A)$ for $i = 1, \cdots, p$ are up to an arbitrary permutation of the eigenvectors of A weighted by 0, $+1$, or -1, i.e., any point $W = [d_{(1)}\phi_{(1)} \ d_{(2)}\phi_{(2)} \ \cdots \ d_{(p)}\phi_{(p)}]$, where $d_{(i)} = 0$, $+1$, or -1, is an equilibrium point of the objective functions $J(\boldsymbol{w}_i; A)$ for $i = 1, \cdots, p$.

Proof: Given in the Appendix.

Theorem 2: Let A1 hold. Then, the points $W^* = [d_1\phi_1 \ d_2\phi_2 \ \cdots \ d_p\phi_p]$, where $d_i = +1$ or -1, are the strict global minimum points of the joint objective functions $J(\boldsymbol{w}_i; A)$ for $i = 1, \cdots, p$. In addition, the points $W = [d_{(1)}\phi_{(1)} \ d_{(2)}\phi_{(2)} \ \cdots \ d_{(p)}\phi_{(p)}]$, where $d_{(i)} = 0$ or $\phi_{(i)} \neq \phi_i$ for $i \in \{1, 2, \cdots, p\}$ are unstable equilibrium points of the objective functions $J(\boldsymbol{w}_i; A)$ for $i = 1, \cdots, p$.

Proof: Given in the Appendix.

We observe that the total number of equilibrium points of the joint objective functions $J(\boldsymbol{w}_i; A)$ for $i = 1, \cdots, p$ are

$$\sum_{i=0}^{p} \frac{2^{p-i} p! n!}{i!(p-i)!(n-p+i)!}.$$

Out of these, there are 2^p global minimum points $W^* = [\pm\phi_1 \ \pm\phi_2 \ \cdots \ \pm\phi_p]$. One of the equilibrium points is the null vector and is a local maximum. The other equilibrium points are aligned to the n eigenvectors of A. Among these, all the points that are not aligned to the principal eigenvector matrix $\Phi_p = [\phi_1 \ \phi_2 \ \cdots \ \phi_p]$ are saddle points, and correspond to larger values of $J(\boldsymbol{w}_i; A)$ for $i = 1, \cdots, p$. The critical points aligned to the principal eigenvector matrix ϕ_p are the global minima.

B. Convergence of Gradient Descent Algorithm (22)

We use Theorem 1 of Ljung [4], [13] for the convergence proof. The theorem deals with nonlinear stochastic algorithms of the form $W_{k+1} = W_k + \eta_k h(W_k, A_k)$, which include (22). Ljung assumes the following: 1) the function $h(W, A_k)$ is continuously differentiable with respect to W and A_k, and the derivatives are, for fixed W and A_k bounded in k and 2) the so called mean vector field $\overline{h}(W) = \lim_{k\to\infty} E[h(W, A_k)]$ exists and is regular; i.e., locally Lipschitz. In order to satisfy these conditions, we make the following assumptions.

Assumption A2: Each A_k is bounded with probability one (w.p. 1), symmetric, real, nonnegative definite, and $\lim_{k\to\infty} E[A_k] = A$ w.p. 1, where A is positive definite.

Assumption A3: $\{\eta_k \in \Re^+\}$ is a decreasing sequence, such that $\sum_{k=0}^{\infty} \eta(k) = \infty$, $\sum_{k=0}^{\infty} \eta_k^r < \infty$ for some $r > 1$, and $\lim_{k\to\infty} \sup(\eta_k^{-1} - \eta_{k-1}^{-1}) < \infty$.

There are several models for generating the random observation sequence $\{A_k\}$. One useful model that is commonly utilized in signal and image processing applications [9], [16], [23] is where $\{A_k\}$ is generated from a sequence of random vectors $\{\boldsymbol{x}_k\}$ as

$$A_k = \beta A_{k-1} + \left(\boldsymbol{x}_k \boldsymbol{x}_k^T - \beta A_{k-1}\right)/k \tag{36}$$

where A_0 is symmetric, and $0 < \beta \leq 1$ is a forgetting factor. If $\{\boldsymbol{x}_k\}$ comes from a stationary process, we choose $\beta = 1$. On the other hand, if $\{\boldsymbol{x}_k\}$ comes from a nonstationary process, we choose $0 < \beta < 1$ to implement an effective window of size $1/(1-\beta)$. This effective window ensures that the past data samples are down-weighted with an exponentially fading window compared to the recent ones in order to afford the tracking capability of the adaptive algorithm. The exact value of β depends on the specific application. Generally speaking, for slow time-varying $\{x_k\}$, β is chosen close to one to implement a large effective window, whereas for fast-time-varying $\{\boldsymbol{x}_k\}$, β is chosen near zero for a small effective window [4].

We next modify the results of Ljung (Theorem 1) [4], [13] to suit algorithm (22) in the following lemma.

Lemma 1: Let A1–A3 hold. Let W^* be a locally asymptotically stable (in the sense of Lyapunov) solution to the ordinary differential equation (ODE)

$$\frac{d}{dt} W(t) = 2AW(t) - W(t)\mathrm{UT}\left[W(t)^T A W(t)\right] - AW(t)\mathrm{UT}\left[W(t)^T W(t)\right] \tag{37}$$

with domain of attraction $D(W^*)$. Then if there is a compact subset S of $D(W^*)$ such that $W_k \in S$ infinitely often, then we have $W_k \to W^*$ with probability one as $k \to \infty$. ■

In Theorems 1 and 2, we have proven that the stable stationary points of the ODE (37) are the principal eigenvectors of A. However, one must, in addition, prove that (22) is stable; i.e., the weight matrix W_k must remain bounded on some realistic conditions. Such boundedness condition is also necessary for W_k to visit a compact subset S of the domain of attraction of W^* infinitely often. It turns out that there exists a uniform upper bound for η_k such that W_k is uniformly bounded. In the following theorem, we determine a uniform upper bound for η_k such that $\boldsymbol{w}_k^1$ is uniformly bounded for all k. Proof for W_k is similar, and will be skipped here.

Theorem 3: Let A1 and A2 hold. Then there exists a uniform upper bound for η_k such that $\boldsymbol{w}_k^1$ is uniformly bounded w.p. 1.

Proof: Given in the Appendix. ■

For a complete convergence proof of algorithm (22), in addition to Theorem 3, we also need to show that W_k visits a compact subset of the domain of attraction of W^* infinitely often. Theoretically, W_k can get stuck in any one of the saddle points. It suffices to say that the presence of saddle points does not cause problems of convergence since they are avoided through random perturbations of W_k in practice, provided η_k satisfies the upper bound in Theorem 3. Since x_k is randomly sampled from a population, matrix A_k generated by (36) is also random. Thus, the gradient descent algorithm (22) randomly deviates from the gradient of $J(\boldsymbol{w}_i; A)$. When W_k is within the domain of attraction of a saddle point, it performs a random walk within this domain. As long as the probability that W_k moves out of this domain is

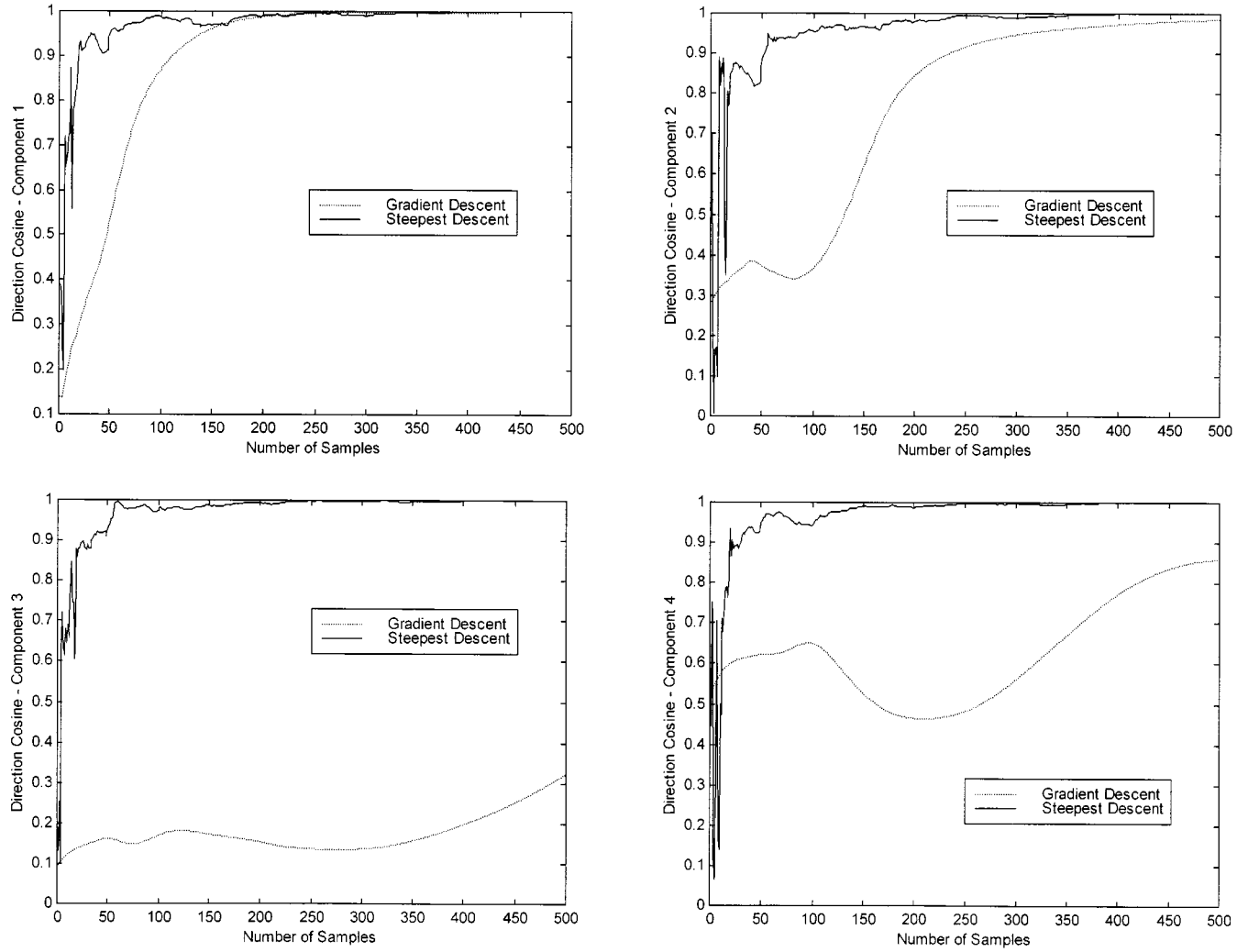

Fig. 1. Convergence of the first four principal eigenvectors of A by the gradient descent (22) and steepest descent (26) algorithms for stationary data.

greater than zero, W_k will eventually reach a global minimum of $J(\boldsymbol{w}_i; A)$.

The convergence of algorithm (22) can now be established by referring to Theorem 1 of Ljung [4], [13].

Theorem 4: Let A1–A3 hold. Assume that with probability one the process $\{W_k\}$ visits infinitely often a compact subset of the domain of attraction of one of the asymptotically stable points $W^* = [\pm\phi_1 \quad \pm\phi_2 \quad \cdots \quad \pm\phi_p]$. Then, with probability one

$$\lim_{k\to\infty} W_k = W^*.$$

Proof: By Theorem 2, $W^* = [\pm\phi_1 \quad \pm\phi_2 \quad \cdots \quad \pm\phi_p]$ are asymptotically stable points of the ODE (37). Since we assume that $\{W_k\}$ visits a compact subset of the domain of attraction of W^* infinitely often, Lemma 1 then implies the theorem. ■

V. Experimental Results

We did two sets of experiments to test the performance of the new PCA algorithms. We did the first set of experiments on *stationary* Gaussian data, and the second set on *nonstationary* Gaussian data. We then compared the steepest descent algorithm against state-of-the-art adaptive PCA algorithms.

A. Experiments with Stationary Data

We generated 500 samples of ten-dimensional Gaussian data with mean zero and covariance given below. Note that this covariance matrix is obtained from the first covariance matrix in [19] multiplied by two. The covariance matrix is shown in (37a) at the bottom of the next page. The eigenvalues of the covariance matrix are

$$11.7996,\ 5.5644,\ 3.4175,\ 2.0589,\ 0.7873,$$
$$0.5878,\ 0.1743,\ 0.1423,\ 0.1213,\ 0.1007.$$

Clearly, the first four eigenvalues are significant and we adaptively compute the corresponding eigenvectors. In order to compute the online data sequence $\{A_k\}$, we generated random data vectors $\{\boldsymbol{x}_k\}$ from the above covariance matrix. We generated

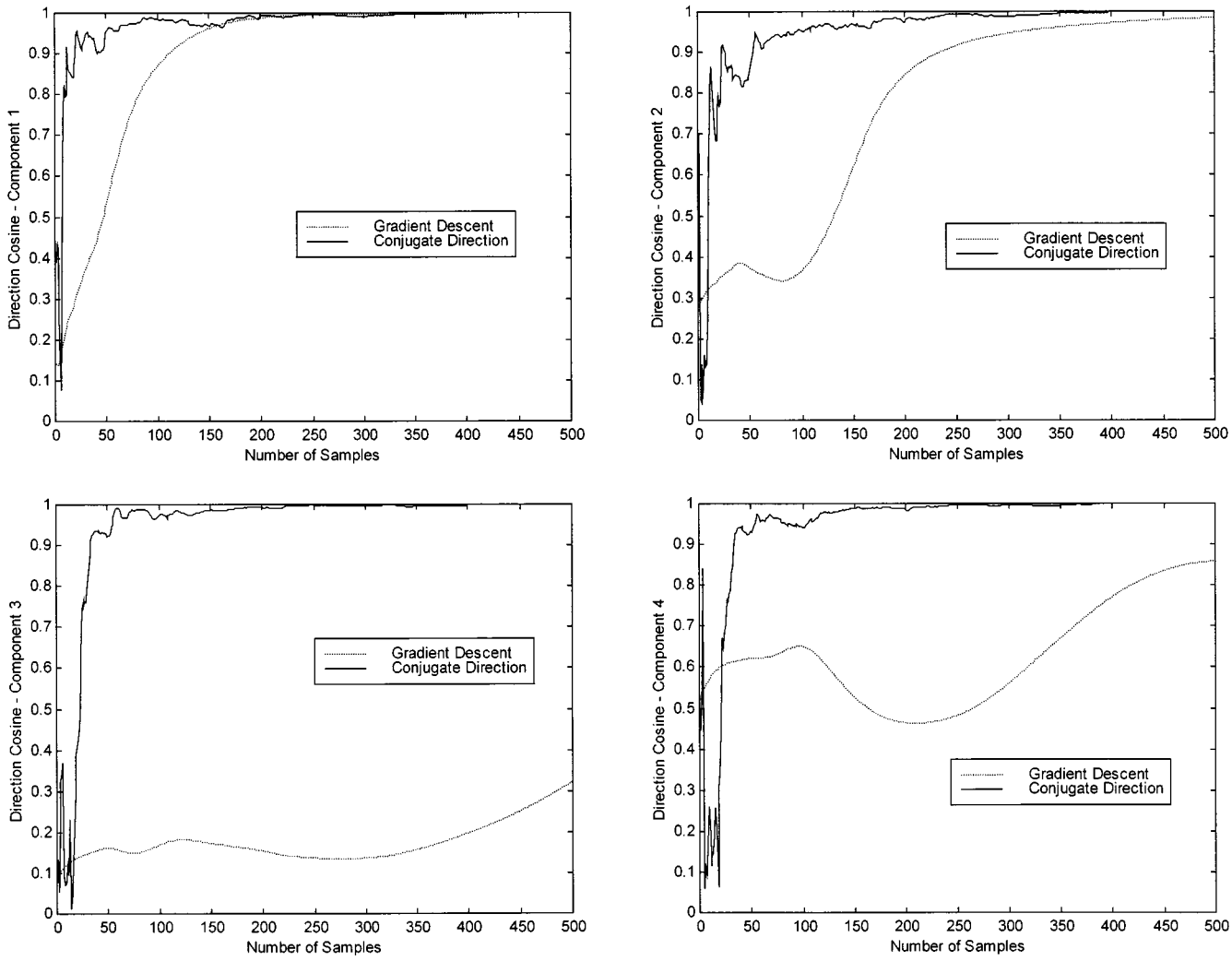

Fig. 2. Convergence of the first four principal eigenvectors of A by the gradient descent (22) and conjugate direction (29) algorithms for stationary data.

$\{A_k\}$ from $\{\boldsymbol{x}_k\}$ by using algorithm (36) with $\beta = 1$. We compute the correlation matrix A after collecting all 500 samples $\boldsymbol{x}_k$ as

$$A = \tfrac{1}{500} \sum_{i=1}^{500} \boldsymbol{x}_i \boldsymbol{x}_i^T.$$

We refer to the eigenvectors and eigenvalues computed from this A by a standard numerical analysis method [11] as the *actual values*.

We used the adaptive gradient descent (22), steepest descent (26), conjugate direction (29), and Newton–Raphson (30) algorithms on the random data sequence $\{A_k\}$. We start all algo-

$$\begin{array}{rrrrrrrrrr}
0.1820 & 0.0760 & -0.1060 & -0.0100 & 0.0200 & -0.2720 & 0.3100 & 0.0600 & 0.0040 & 0.0640 \\
0.0760 & 0.7460 & 0.0360 & -0.0560 & -0.0220 & -0.7340 & 0.3080 & -0.1140 & -0.0620 & -0.1300 \\
-0.1060 & 0.0360 & 2.8600 & 0.0340 & 0.1100 & -0.9000 & -0.0760 & -0.5960 & -0.0820 & -0.0600 \\
-0.0100 & -0.0560 & 0.0340 & 0.1680 & -0.0100 & 0.0320 & 0.0840 & -0.0440 & 0.0020 & 0.0100 \\
0.0200 & -0.0220 & 0.1100 & -0.0100 & 0.1420 & 0.1760 & 0.1160 & -0.1380 & -0.0160 & 0.0060 \\
-0.2720 & -0.7340 & -0.9000 & 0.0320 & 0.1760 & 11.4400 & -1.0880 & -0.4960 & 0.0100 & 0.1900 \\
0.3100 & 0.3080 & -0.0760 & 0.0840 & 0.1160 & -1.0880 & 5.5000 & -0.6860 & -0.0220 & -0.2400 \\
0.0600 & -0.1140 & -0.5960 & -0.0440 & -0.1380 & -0.4960 & -0.6860 & 2.9000 & 0.1560 & 0.0560 \\
0.0040 & -0.0620 & -0.0820 & 0.0020 & -0.0160 & 0.0100 & -0.0220 & 0.1560 & 0.1340 & 0.0300 \\
0.0640 & -0.1300 & -0.0600 & 0.0100 & 0.0060 & 0.1900 & -0.2400 & 0.0560 & 0.0300 & 0.6820
\end{array} \tag{37a}$$

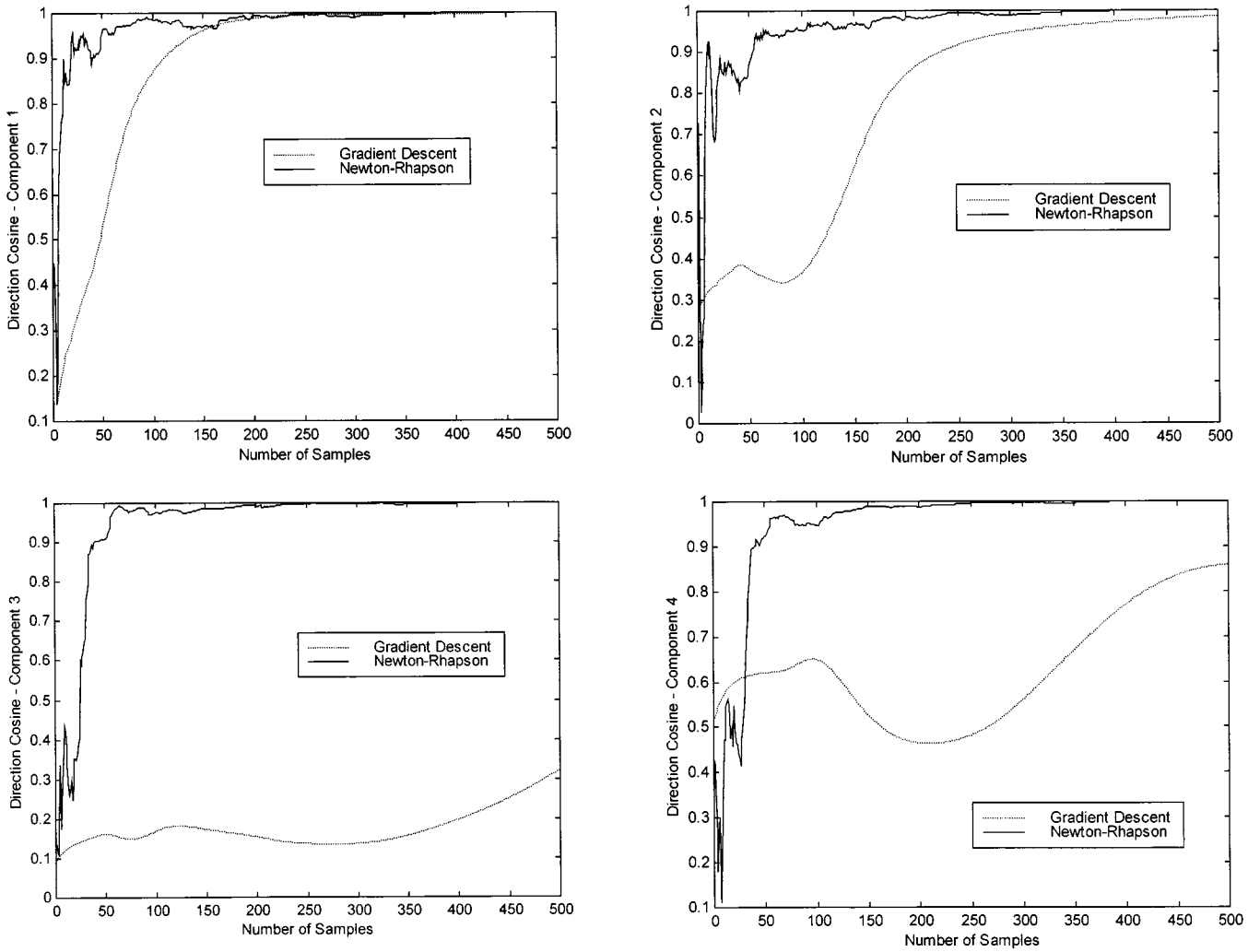

Fig. 3. Convergence of the first four principal eigenvectors of A by the gradient descent (22) and Newton–Raphson (30) algorithms for stationary data.

rithms with $W_0 = 0.1 * \text{ONE}$, where ONE is an $n \times p$ matrix whose all elements are ones. In order to measure the convergence and accuracy of the algorithms, we computed the direction cosine at kth update of each adaptive algorithm as

$$\text{Direction Cosine } (k) = \frac{\left| \boldsymbol{w}_k^{i^T} \phi_i \right|}{\|\boldsymbol{w}_k^i\| \, \|\phi_i\|} \tag{38}$$

where $\boldsymbol{w}_k^i$ is the estimated eigenvector of A_k at kth update, and ϕ_i is the actual eigenvector computed from all collected samples by a conventional numerical analysis method.

Figs. 1–4 show the iterates of the four algorithms to compute the first four principal eigenvectors of A. For the gradient descent (22) algorithm, we used $\eta_k = 1/(400 + k)$. For the conjugate direction method, we used the Hestenes–Stiefel method to compute β_k^i. For the steepest descent, conjugate direction, and Newton–Raphson methods, we chose α_k^i by solving a cubic equation as described in Sections III-B–III-D, respectively.

It is clear from Figs. 1–4, that the steepest descent, conjugate direction, and Newton–Raphson algorithms converge faster than the gradient descent algorithm in spite of a careful selection of η_k for the gradient descent algorithm. Besides, the new algorithms do not require *ad hoc* selections of η_k. Instead, the gain parameters α_k^i and β_k^i are computed from the online data sequence.

Comparison between the four algorithms in Fig. 4, show small differences between them for the first four principal eigenvectors of A. Among the three faster converging algorithms, the steepest descent algorithm (26) requires the smallest amount computation per iteration. Therefore, these limited experiments show that the steepest descent adaptive algorithm (26) is most suitable for optimum speed and computation among the four algorithms presented here.

B. Experiments with Nonstationary Data

In order to demonstrate the tracking ability of the algorithms with nonstationary data, we generated 500 samples of zero-mean ten-dimensional Gaussian data with the covariance matrix stated before. We then drastically changed the data sequence by generating 1000 samples of zero-mean ten-dimensional Gaussian data with the covariance matrix below (from

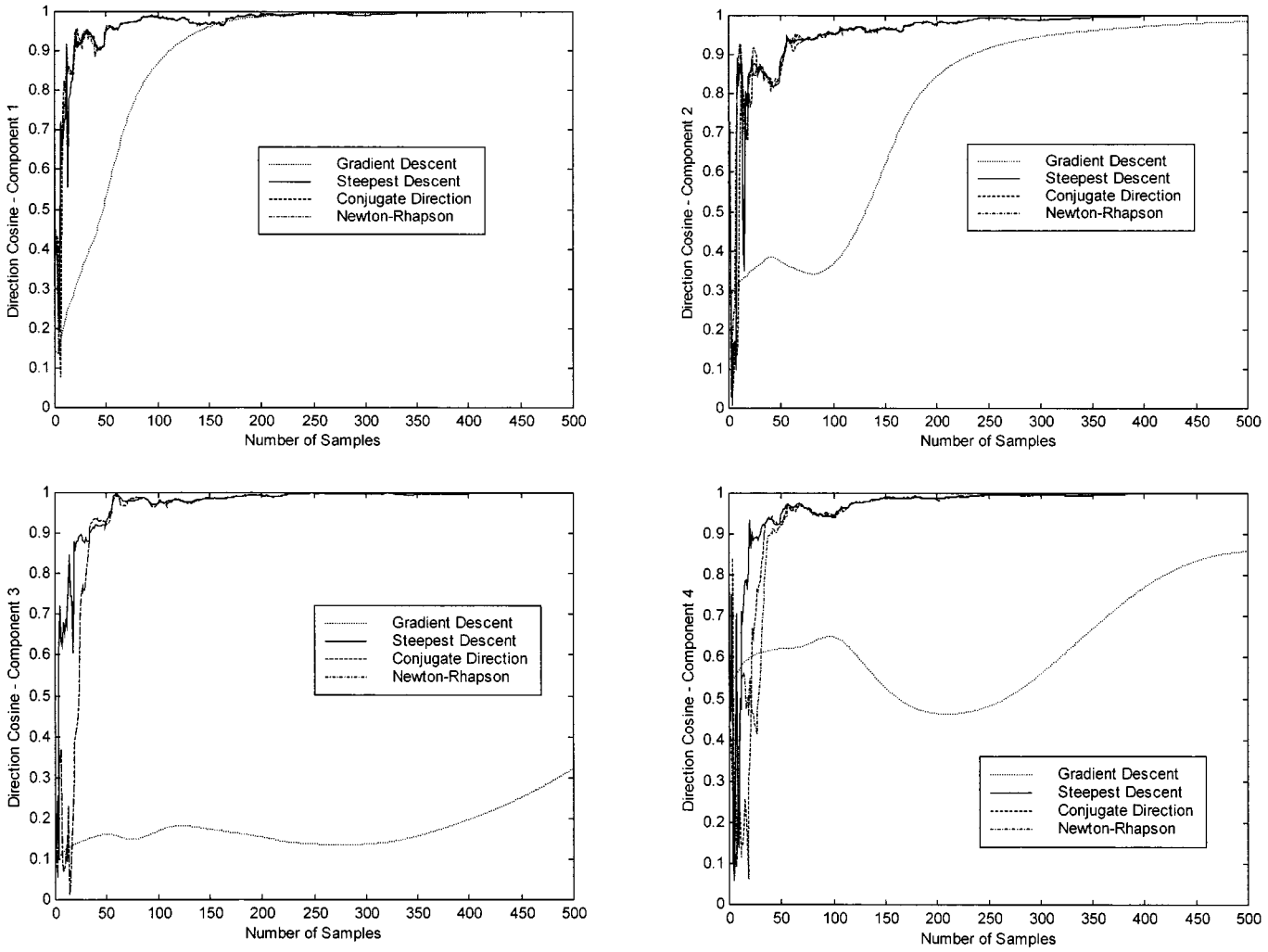

Fig. 4. Convergence of the first four principal eigenvectors of A by the gradient descent (22), steepest descent (26), conjugate direction (29), and Newton–Raphson (30) algorithms for stationary data.

[19]) as shown in (38a) at the bottom of the next page. The eigenvalues of this covariance matrix are

$$23.3662,\ 16.5698,\ 6.8611,\ 1.8379,\ 1.5452,$$
$$0.7010,\ 0.3851,\ 0.3101,\ 0.2677,\ 0.2278$$

which are drastically different from the previous eigenvalues. We generated $\{A_k\}$ from $\{\boldsymbol{x}_k\}$ by using algorithm (36) with $\beta = 0.995$. We used the adaptive gradient descent (22), steepest descent (26), conjugate direction (29), and Newton–Raphson (30) algorithms on the random observation sequence $\{A_k\}$, and measured the convergence accuracy of the algorithms by computing the direction cosine at kth update of each adaptive algorithm as shown in (38). We start all algorithms with $W_0 = 0.1 * \text{ONE}$, where ONE is an $n \times p$ matrix whose all elements are ones.

Figs. 5–8 show the iterates of the four algorithms to compute the first four principal eigenvectors of the two covariance matrices described before. For the gradient descent (22) algorithm, we used $\eta_k = 1/(400+k)$. For the conjugate direction method, we used the Hestenes–Stiefel method to compute β_k^i. For the steepest descent, conjugate direction and Newton–Raphson methods, we chose α_k^i by solving a cubic equation as described in Sections III-B–III-D, respectively.

Once again, it is clear from Figs. 5–8, that the steepest descent, conjugate direction, and Newton–Raphson algorithms converge faster and track the changes in data much better than the traditional gradient descent algorithm. In some cases, such as Fig. 6 for the third principal eigenvector, the gradient descent algorithm fails as the data sequence changes, but the new algorithms perform correctly.

Comparison between the four algorithms in Fig. 8, show small differences between them for the first four principal eigenvectors. Once again, among the three faster converging algorithms, since the steepest descent algorithm (26) requires the smallest amount computation per iteration, it is most suitable for optimum speed and computation.

C. Comparison with State-of-the-Art Algorithms

We compared our steepest descent algorithm (26) with Yang's projection approximation subspace tracking (PASTd) algorithm [23], Bannour and Sadjadi's RLS algorithm [3], and

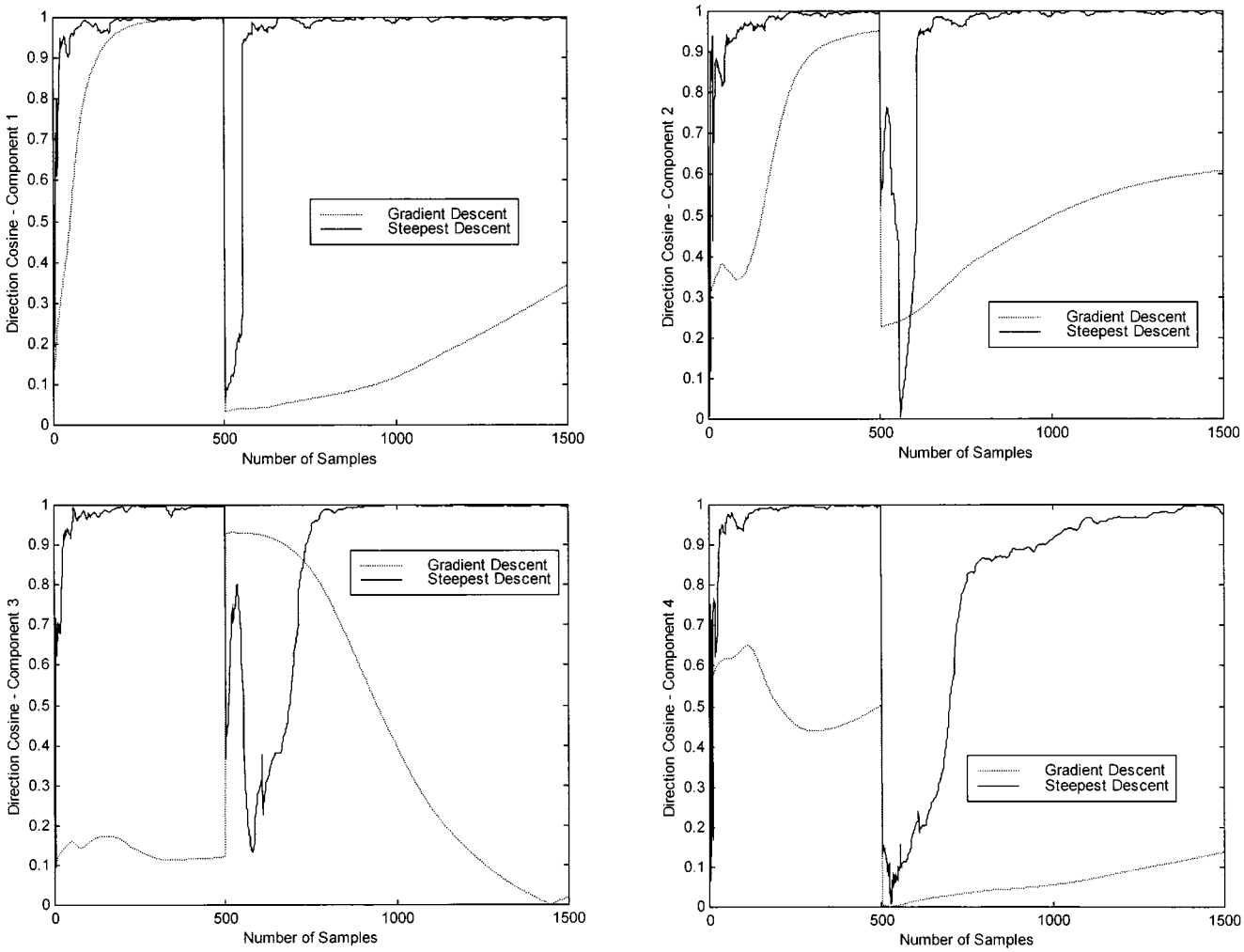

Fig. 5. Convergence of the first four principal eigenvectors of two covariance matrices by the gradient descent (22) and steepest descent (26) algorithms for nonstationary data.

Fu and Dowling's CGET1 algorithm [9], [10]. We first tested the four algorithms on the stationary data described in Section V-A.

We define ONE as an $n \times p$ ($p \leq n$) matrix whose all elements are ones. The initial values for each algorithm are as follows:

Our steepest descent algorithm

$$W_0 = 0.1 * \text{ONE}.$$

$$\begin{array}{rrrrrrrrrr}
0.3600 & 0.0040 & -0.0320 & -0.7640 & -0.0280 & 0.1640 & -0.1200 & -0.2320 & 0.0880 & 0.1280 \\
0.0040 & 0.3680 & -0.0440 & 0.0320 & -0.0560 & -0.0080 & 0.0480 & -0.0400 & -0.0840 & -0.0080 \\
-0.0320 & -0.0440 & 0.3280 & 0.3280 & 0.0560 & -0.0800 & -0.2320 & 0.4200 & 0.0160 & 0.0920 \\
-0.7640 & 0.0320 & 0.3280 & 22.7200 & -0.3840 & -0.0600 & 2.5840 & 0.8760 & -0.9520 & 0.8720 \\
-0.0280 & -0.0560 & 0.0560 & -0.3840 & 0.3040 & -0.1400 & -0.1600 & -0.0920 & 0.1080 & -0.0560 \\
0.1640 & -0.0080 & -0.0800 & -0.0600 & -0.1400 & 1.8320 & 0.5520 & -1.0040 & 0.0480 & 0.1560 \\
-0.1200 & 0.0480 & -0.2320 & 2.5840 & -0.1600 & 0.5520 & 7.2800 & -0.7320 & -0.0080 & 0.4680 \\
-0.2320 & -0.0400 & 0.4200 & 0.8760 & -0.0920 & -1.0040 & -0.7320 & 16.2800 & -1.8560 & 0.5880 \\
0.0880 & -0.0840 & 0.0160 & -0.9520 & 0.1080 & 0.0480 & -0.0080 & -1.8560 & 1.0520 & 0.2160 \\
0.1280 & -0.0080 & 0.0920 & 0.8720 & -0.0560 & 0.1560 & 0.4680 & 0.5880 & 0.2160 & 1.5480
\end{array} \tag{38a}$$

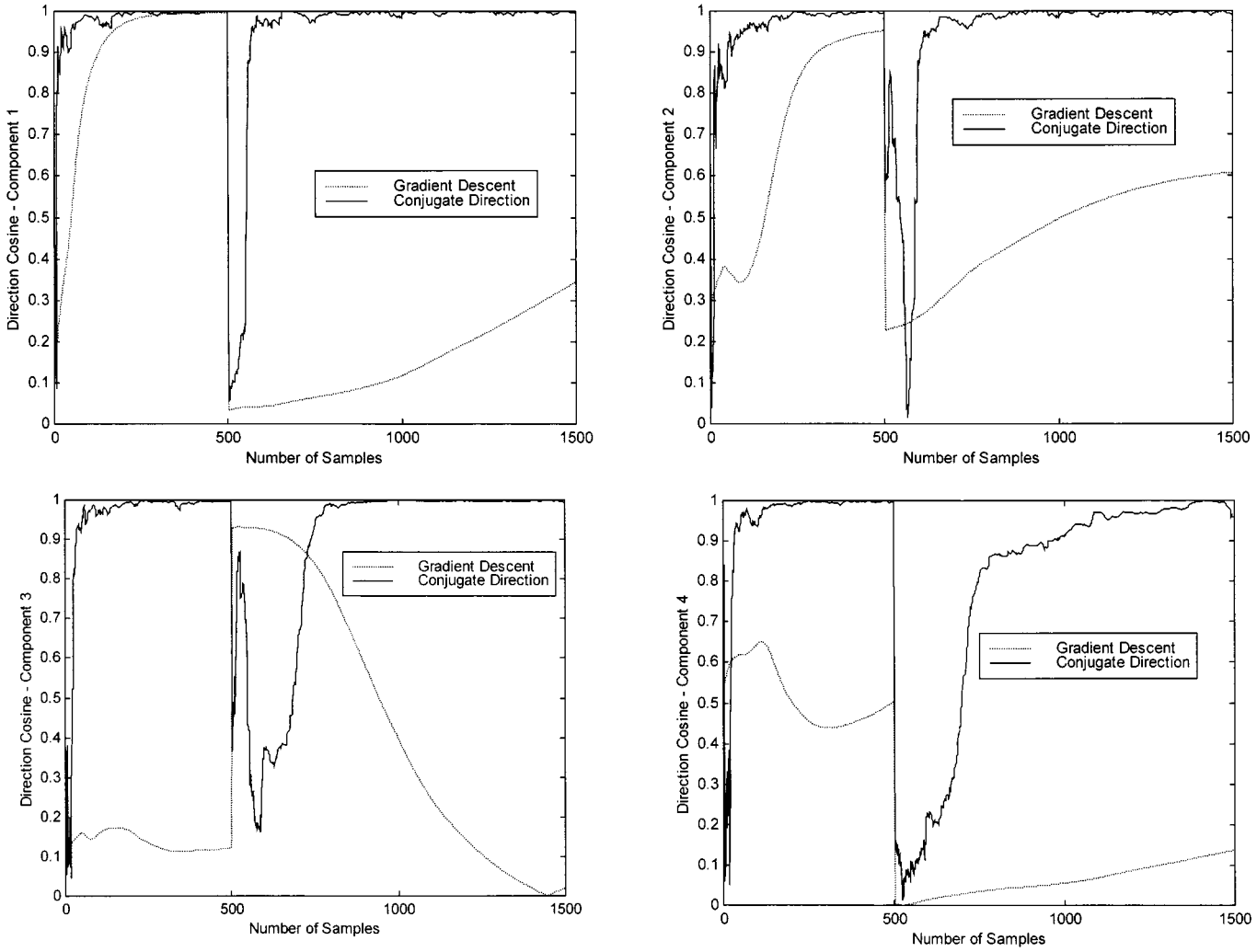

Fig. 6. Convergence of the first four principal eigenvectors of two covariance matrices by the gradient descent (22) and conjugate direction (29) algorithms for nonstationary data.

Yang's projection approximation subspace tracking (PASTd) algorithm

$W_0 = 0.1 * \text{ONE}$ and $d_0^i = 0.2$ for $i = 1, 2, \cdots, p$ $(p \leq n)$.

Bannour and Sadjadi's RLS algorithm

$$W_0 = 0.1 * \text{ONE} \text{ and } P_0 = \text{ONE}.$$

Fu and Dowling's conjugate gradient eigenstructure tracking (CGET1) algorithm

$$W_0 = 0.1 * \text{ONE} \text{ and } A_0 = \boldsymbol{x}_k \boldsymbol{x}_k^T.$$

We found that the performance of the PASTd and RLS algorithms depended considerably on the initial choices of d_0^i and P_0, respectively. We, therefore, chose the initial values that gave us the best results for most experiments. The results of this experiment are shown in Fig. 9.

We observe, from Fig. 9 that the steepest descent and CGET1 algorithms perform quite well for all four principal eigenvectors. The RLS performed a little better than the PASTd algorithm for the minor eigenvectors. For the major eigenvectors, all algorithms performed well. The differences between the algorithms were evident for the minor (third and fourth) eigenvectors.

We next applied the four algorithms on nonstationary data described in Section V-B with $\beta = 0.995$ in (36). The results of this experiment are shown in Fig. 10.

We observe that the steepest descent and CGET1 algorithms perform quite well for all four principal eigenvectors. The PASTd algorithm performs better than the RLS algorithm in handling nonstationarity. This is expected since the PASTd algorithm accounts for nonstationarity with a forgetting factor of $\beta = 0.995$, whereas the RLS algorithm has no such option.

VI. Concluding Remarks

We present an unconstrained objective function to obtain various new adaptive algorithms for PCA by using nonlinear optimization methods such as gradient descent (GD), steepest descent (SD), conjugate gradient (CG), and NR. We analyze the landscape of the unconstrained cost function. Convergence analysis of the GD adaptive algorithm is also presented. Comparison among these algorithms with stationary and nonstationary data show that the SD, CG, and NR algorithms have faster tracking abilities compared to the GD algorithm.

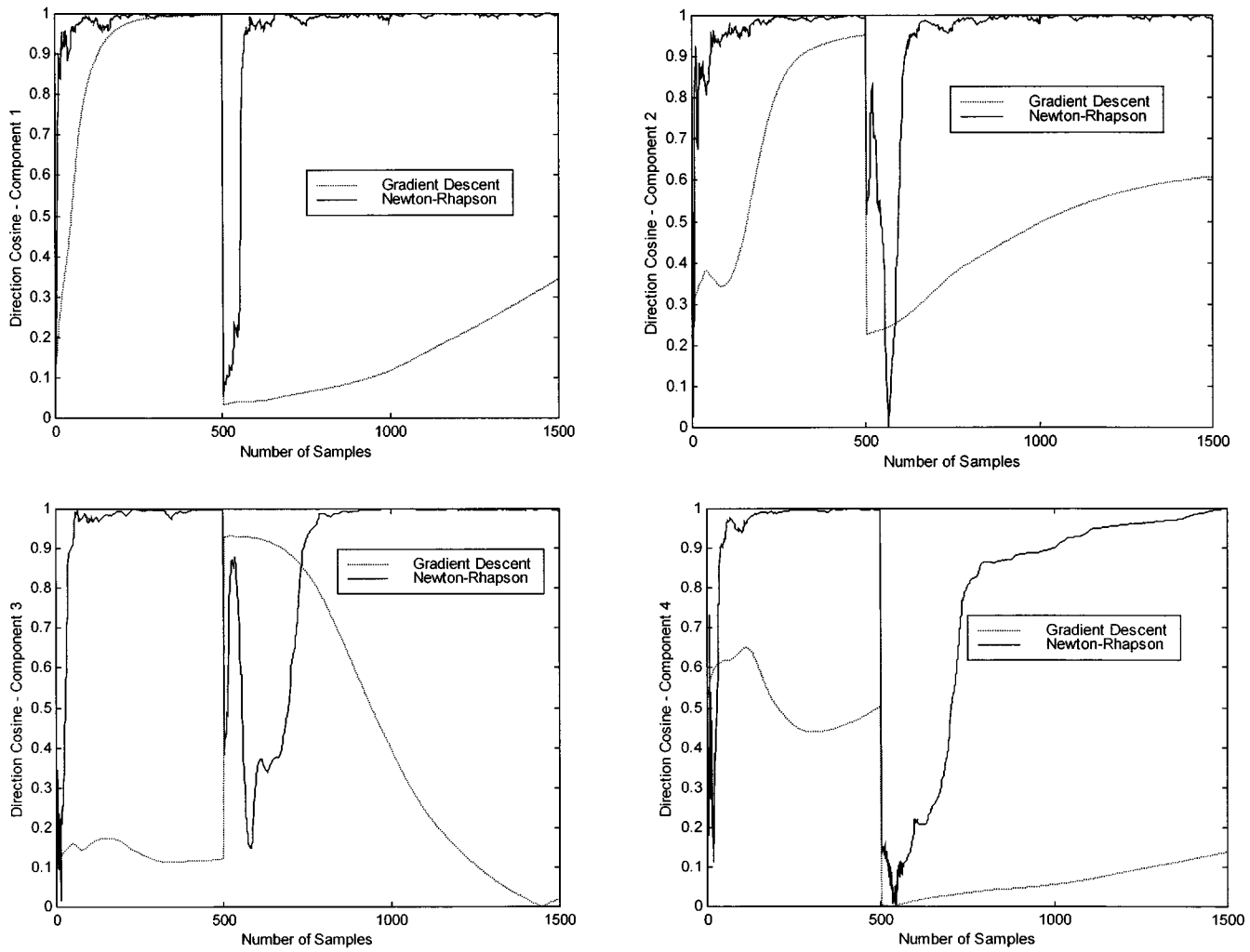

Fig. 7. Convergence of the first four principal eigenvectors of two covariance matrices by the gradient descent (22) and Newton–Raphson (30) algorithms for nonstationary data.

Further consideration should be given to the computational complexity of the algorithms. SD, CG, and NR algorithms have computational complexity of $O(pn^2)$. If, however, we use the estimate $A_k = \boldsymbol{x}_k\boldsymbol{x}_k^T$ instead of (36) in the GD algorithm, then the computational complexity drops to $O(pn)$, although the convergence gets slower. The CGET1 algorithm has complexity $O(pn^2)$. The PASTd and RLS algorithms have complexity $O(pn)$. However, their convergence is slower than the SD and CGET1 algorithms as shown in Figs. 9 and 10. Further note that our GD algorithm can be implemented by parallel architecture as shown by the examples in [8].

APPENDIX

A. Computation of α_k^i for Steepest Descent

From $J(\boldsymbol{w}_i;\ A)$ in (12), we compute α that minimizes $J(\boldsymbol{w}_i - \alpha\boldsymbol{g}_i;\ A)$, where

$$\boldsymbol{g}_i = -2A\boldsymbol{w}_i + \boldsymbol{w}_i\boldsymbol{w}_i^TA\boldsymbol{w}_i + \sum_{j=1}^{i-1}\boldsymbol{w}_j\boldsymbol{w}_j^TA\boldsymbol{w}_i + A\boldsymbol{w}_i\boldsymbol{w}_i^T\boldsymbol{w}_i + \sum_{j=1}^{i-1}A\boldsymbol{w}_j\boldsymbol{w}_j^T\boldsymbol{w}_i. \quad \text{(A.1)}$$

We have

$$\frac{dJ(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i)}{d\alpha} = \frac{1}{2}\operatorname{tr}\left[\nabla_{\boldsymbol{w}_i-\alpha\boldsymbol{g}_i}J(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i)\frac{d(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i)^T}{d\alpha}\right] = -\tfrac{1}{2}\boldsymbol{g}_i^T\nabla_{\boldsymbol{w}_i-\alpha\boldsymbol{g}_i}J(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i) \quad \text{(A.2)}$$

where

$$\begin{aligned}(1/2)&\nabla_{\boldsymbol{w}_i-\alpha\boldsymbol{g}_i}J(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i)\\ &= -2A(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i) + (\boldsymbol{w}_i-\alpha\boldsymbol{g}_i)(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i)^TA(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i)\\ &\quad + A(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i)(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i)^T(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i)\\ &\quad + \sum_{j=1}^{i-1}\boldsymbol{w}_j\boldsymbol{w}_j^TA(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i) + \sum_{j=1}^{i-1}A\boldsymbol{w}_j\boldsymbol{w}_j^T(\boldsymbol{w}_i-\alpha\boldsymbol{g}_i).\end{aligned}$$

Simplifying (A.2), we obtain the cubic equation below

$$c_3\alpha^3 + c_2\alpha^2 + c_1\alpha + c_0 = 0$$

where

$$\begin{aligned}c_0 &= -\boldsymbol{g}_i^T\boldsymbol{g}_i;\\ c_1 &= \boldsymbol{g}_i^TH_i\boldsymbol{g}_i;\\ c_2 &= -3\left(\boldsymbol{g}_i^TA\boldsymbol{g}_i\boldsymbol{w}_i^T\boldsymbol{g}_i + \boldsymbol{w}_i^TA\boldsymbol{g}_i\boldsymbol{g}_i^T\boldsymbol{g}_i\right);\\ c_3 &= 2\boldsymbol{g}_i^TA\boldsymbol{g}_i\boldsymbol{g}_i^T\boldsymbol{g}_i.\end{aligned}$$

Here H_i is the Hessian of $J(\boldsymbol{w}_i;\ A)$ given in (25).

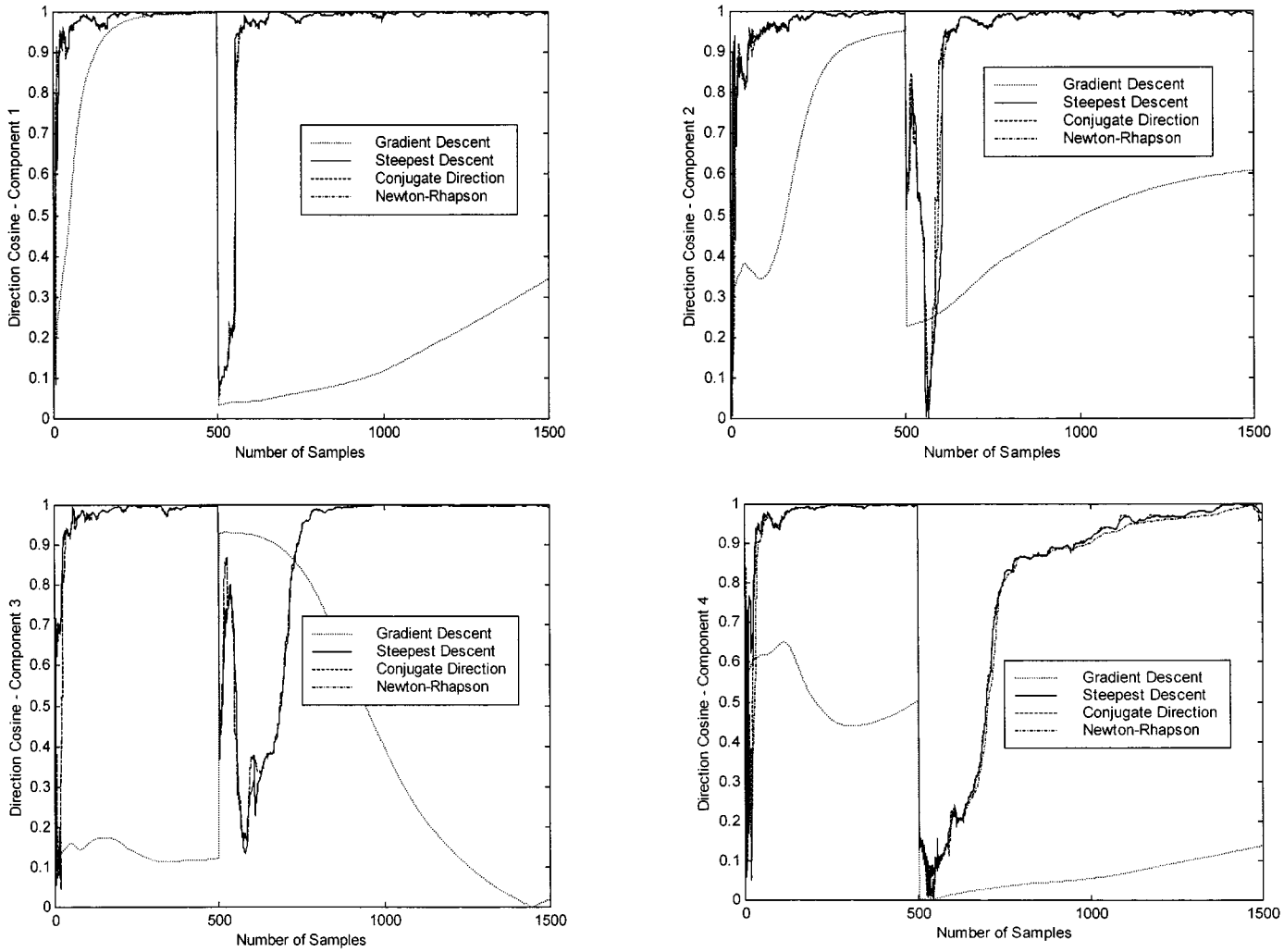

Fig. 8. Convergence of the first four principal eigenvectors of two covariance matrices by the gradient descent (22), steepest descent (26), conjugate direction (29), and Newton–Raphson (30) algorithms for nonstationary data.

It is well known that a cubic polynomial has at least one real root. The roots can also be computed in closed form as shown in [1]. Expressing $J(\boldsymbol{w}_i - \alpha \boldsymbol{g}_i;\, A)$ in terms of α, we obtain

$$\begin{aligned}\Delta J &= J(\boldsymbol{w}_i - \alpha \boldsymbol{g}_i;\, A) - J(\boldsymbol{w}_i;\, A) \\ &= \frac{c_3\alpha^4}{2} + \frac{2c_2\alpha^3}{3} + c_1\alpha^2 + 2c_0\alpha.\end{aligned}$$

If a root is complex, then ΔJ and $\boldsymbol{w}_i - \alpha \boldsymbol{g}_i$ are complex, and clearly, this is not the root we are looking for. If we have three real roots, then we can either take the root corresponding to minimum $J(\boldsymbol{w}_i - \alpha \boldsymbol{g}_i;\, A)$ or the one corresponding to $3c_3\alpha^2 + 2c_2\alpha + c_1 > 0$.

B. Proof of Theorem 1

In order to satisfy the first-order conditions for the existence of the equilibrium points of the joint objective functions $J(\boldsymbol{w}_i;\, A)$ for $i = 1, \cdots, p$, we need to find a $W^* = [\boldsymbol{w}_1^* \;\; \boldsymbol{w}_2^* \;\; \cdots \;\; \boldsymbol{w}_p^*]$ such that

$$\boldsymbol{g}_{ij} = (1/2)\nabla_{\boldsymbol{w}_j} J(\boldsymbol{w}_i^*;\, A) = 0 \quad \text{for } i, j = 1, \cdots, p. \tag{A.3}$$

We define a matrix $W = [\boldsymbol{w}_1, \cdots, \boldsymbol{w}_p]$ $(p \le n)$, for which the columns are the p weight vectors that converge to the p principal eigenvectors of A, respectively. We first find the equilibrium points of the diagonal elements of the Jacobian described in (A.3). We prove the theorem by induction.

From $J(\boldsymbol{w}_1;\, A)$ in (12) we have

$$\begin{aligned}\boldsymbol{g}_{11} &= (1/2)\nabla_{\boldsymbol{w}_1} J(\boldsymbol{w}_1;\, A) \\ &= -2A\boldsymbol{w}_1 + \boldsymbol{w}_1\boldsymbol{w}_1^T A\boldsymbol{w}_1 + A\boldsymbol{w}_1\boldsymbol{w}_1^T\boldsymbol{w}_1.\end{aligned} \tag{A.4}$$

We expand $\boldsymbol{w}_1$ in terms of the entire orthonormal set of eigenvectors of A as

$$\boldsymbol{w}_1 = \sum_{r=1}^{n} a_r \phi_r. \tag{A.5}$$

Substituting this $\boldsymbol{w}_1$ into (A.4), and multiplying it to the left by ϕ_k^T $(k = 1, \cdots, n)$, and then equating it to zero, we get

$$a_k\left(-2\lambda_k + \sum_{r=1}^{n} a_r^2(\lambda_r + \lambda_k)\right) = 0 \tag{A.6}$$

for $k = 1, \cdots, n$, which gives us

$$a_k = 0 \tag{A.7}$$

or

$$\sum_{r=1}^{n} a_r^2(\lambda_r + \lambda_k) = 2\lambda_k \tag{A.8}$$

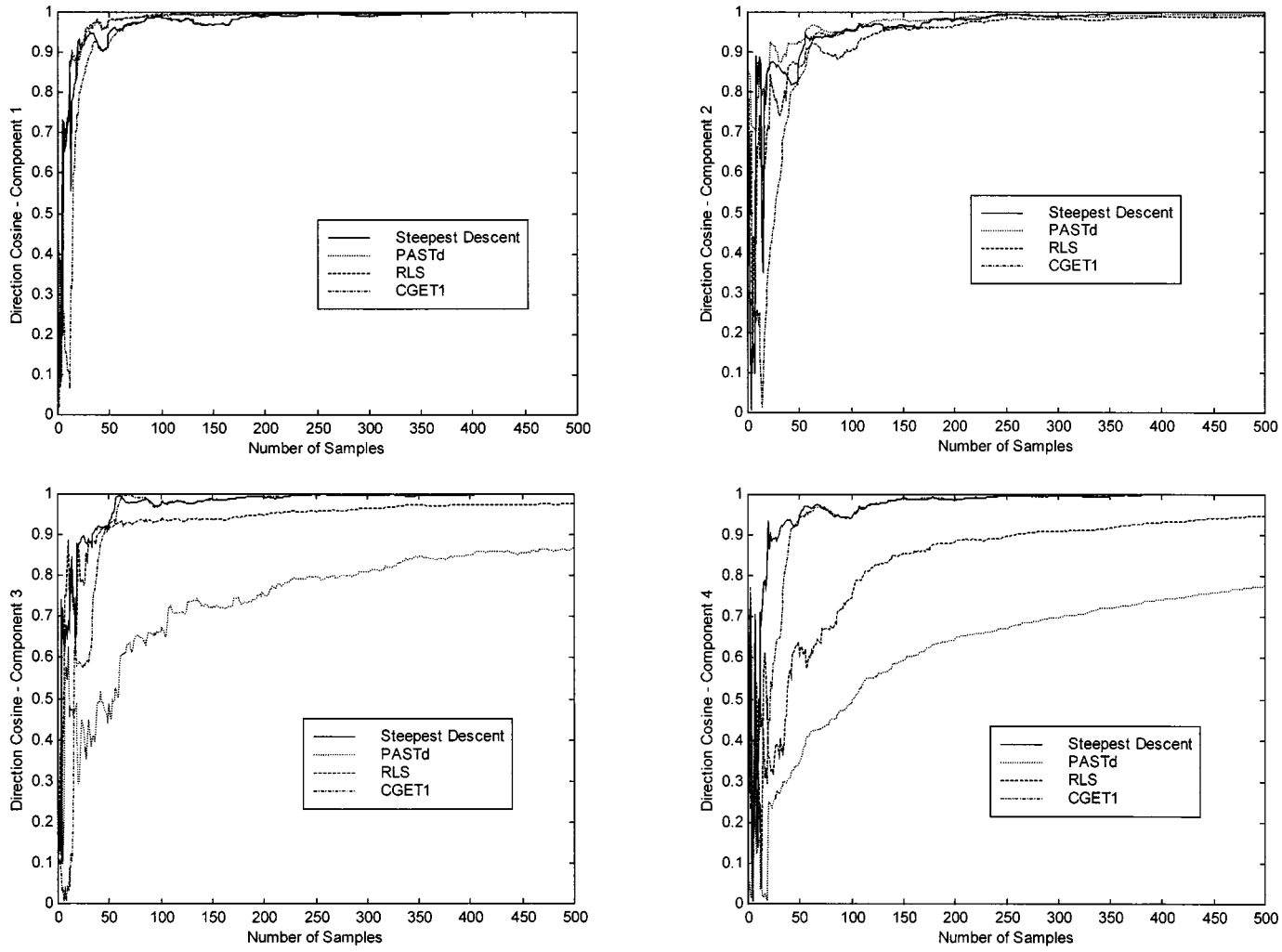

Fig. 9. Convergence of the first four principal eigenvectors of A by our steepest descent (26), Yang's PASTd, Bannour–Sadjadi's RLS, and Fu–Dowling's CGET1 algorithms for stationary data.

for $k = 1, \cdots, n$. Rewriting (A.8), we obtain

$$\sum_{r=1, r\neq k}^{n} a_r^2(\lambda_r + \lambda_k) + 2(a_k^2 - 1)\lambda_k = 0. \tag{A.9}$$

Since A is positive definite and since each eigenvalue of A is of unit multiplicity, from (A.9) we obtain $a_r = 0$ for $r = 1, \cdots, n$, $r \neq k$, and $a_k = \pm 1$ for $k = 1, \cdots, n$. Clearly, there can be at most one nonzero $a_k = \pm 1$. Thus, the equilibrium points of $J(\boldsymbol{w}_1; A)$ are

$$\boldsymbol{w}_1 = d_{(1)}\phi_{(1)}$$

where $\phi_{(1)}$ is a permutation of the eigenvectors $\phi_1, \cdots, \phi_n$, and $d_{(1)} = 0, +1$, or -1.

We next consider the equilibrium points of $J(\boldsymbol{w}_2; A)$, whose gradient with respect to $\boldsymbol{w}_2$ is

$$\begin{aligned}\boldsymbol{g}_{22} &= (1/2)\nabla_{\boldsymbol{w}_2} J(\boldsymbol{w}_2; A) \\ &= -2A\boldsymbol{w}_2 + \boldsymbol{w}_2\boldsymbol{w}_2^T A\boldsymbol{w}_2 + A\boldsymbol{w}_2\boldsymbol{w}_2^T\boldsymbol{w}_2 \\ &\quad + \boldsymbol{w}_1\boldsymbol{w}_1^T A\boldsymbol{w}_2 + A\boldsymbol{w}_1\boldsymbol{w}_1^T\boldsymbol{w}_2. \end{aligned} \tag{A.10}$$

By substituting $\boldsymbol{w}_2 = \sum_{r=1}^{n} a_r\phi_r$ into (A.10), and multiplying it to the left by ϕ_k^T, and then equating it to zero, we get

$$-2a_k\lambda_k + a_k\sum_{r=1}^{n} a_r^2(\lambda_r + \lambda_k) + 2d_{(1)}^2\lambda_{(1)}a_{(1)}\delta_{k,(1)} = 0 \tag{A.11}$$

for $k = 1, \cdots, n$. Here $\delta_{i,j}$ is the Kronecker's delta. We have the following two conditions:

1) If $k = (1)$, then we obtain from (A.11)
$$a_{(1)} = 0 \tag{A.12}$$
or
$$\sum_{r=1}^{n} a_r^2(\lambda_r + \lambda_{(1)}) = -2d_{(1)}^2\lambda_{(1)} + 2\lambda_{(1)}. \tag{A.13}$$
If $d_{(1)} = \pm 1$, then we obtain from (A.13) that all $a_r = 0$ for $r = 1, \cdots, n$. On the other hand, if $d_{(1)} = 0$ then we obtain an equation that is similar to (A.8). We conclude that $a_{(1)} = \pm 1$ and all other $a_r = 0$.
2) If $k \neq (1)$, then we obtain an equation that is same as (A.6), from which we conclude that $a_k = 0$, or there is at most one nonzero $a_k = \pm 1$.

Thus, the equilibrium points of $J(\boldsymbol{w}_2; A)$ are

$$\boldsymbol{w}_2 = d_{(2)}\phi_{(2)} \tag{A.14}$$

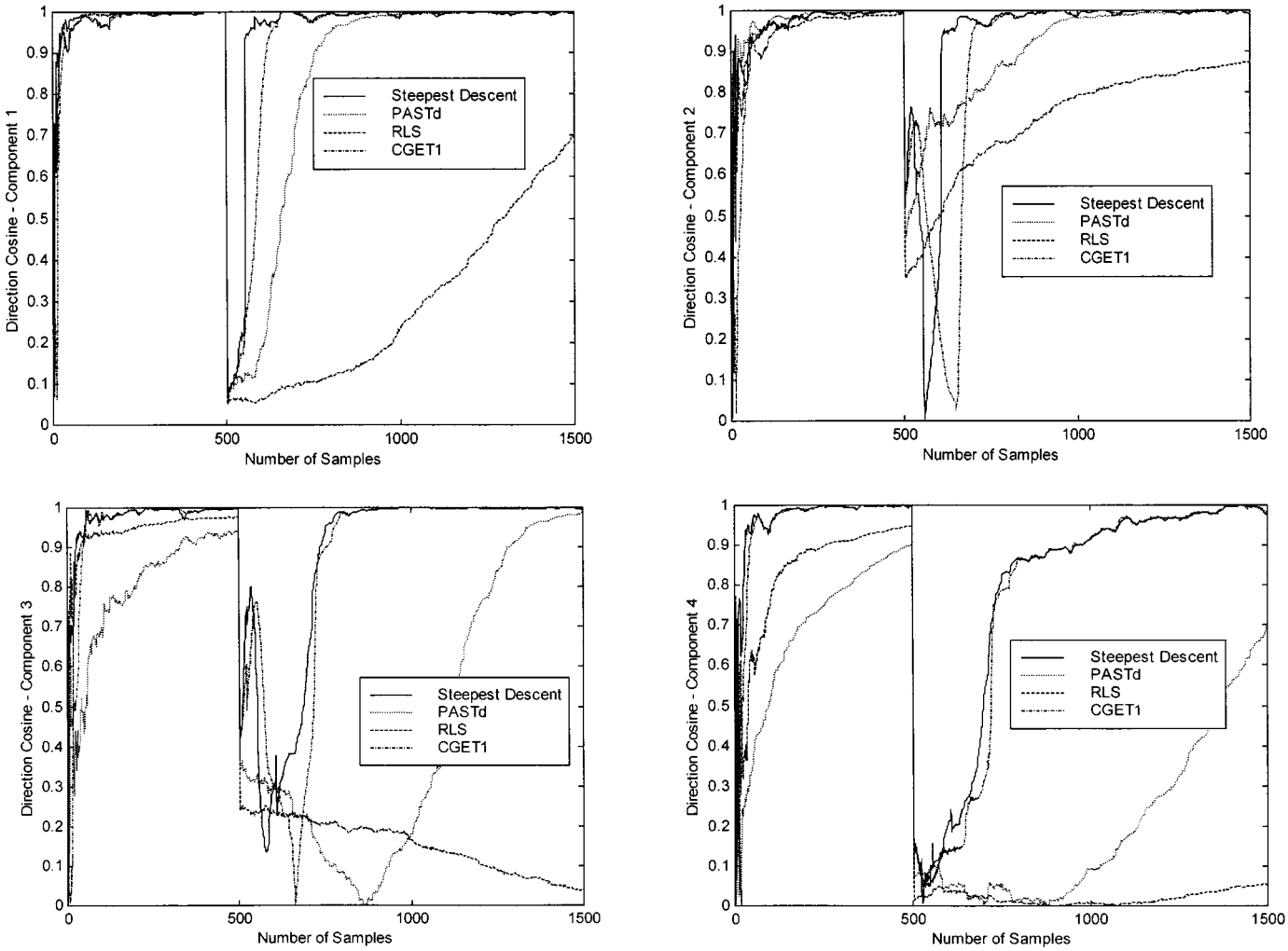

Fig. 10. Convergence of the first four principal eigenvectors of two covariance matrices by our steepest descent (26), Yang's PASTd, Bannour–Sadjadi's RLS, and Fu–Dowling's CGET1 algorithms for nonstationary data.

where $\phi_{(2)}$ is a permutation of the eigenvectors $\phi_1, \cdots, \phi_n$, and $d_{(2)} = 0, +1$, or -1. Furthermore, if $d_{(1)} = 0$, then $\phi_{(2)}$ can be $\phi_{(1)}$.

We now assume that the equilibrium points of $J(\boldsymbol{w}_{i1}; A)$ are

$$\boldsymbol{w}_{i-1} = d_{(i-1)}\phi_{(i-1)} \tag{A.15}$$

where $\phi_{(j)}$ is a permutation of the eigenvectors $\phi_1, \cdots, \phi_n$, and $d_{(j)} = 0, +1$, or -1. Furthermore, for any $j < i-1$, if $d_{(j)} = 0$ then $\phi_{(i-1)}$ can be $\phi_{(j)}$. We now consider the equilibrium points of $J(\boldsymbol{w}_i; A)$, whose gradient with respect to $\boldsymbol{w}_i$ is

$$\begin{aligned}\boldsymbol{g}_{ii} &= (1/2)\nabla_{\boldsymbol{w}_i} J(\boldsymbol{w}_i; A) \\ &= -2A\boldsymbol{w}_i + \boldsymbol{w}_i\boldsymbol{w}_i^T A\boldsymbol{w}_i + A\boldsymbol{w}_i\boldsymbol{w}_i^T\boldsymbol{w}_i \\ &\quad + \sum_{j=1}^{i-1} \boldsymbol{w}_j\boldsymbol{w}_j^T A\boldsymbol{w}_i + \sum_{j=1}^{i-1} A\boldsymbol{w}_j\boldsymbol{w}_j^T\boldsymbol{w}_i. \end{aligned} \tag{A.16}$$

By substituting $\boldsymbol{w}_i = \sum_{r=1}^{n} a_r\phi_r$ into (A.16), and multiplying it to the left by ϕ_k^T, and then equating it to zero, we get

$$-2a_k\lambda_k + a_k \sum_{r=1}^{n} a_r^2(\lambda_r + \lambda_k) + 2d_{(j)}^2\lambda_{(j)}a_{(j)}\delta_{k,(j)} = 0 \tag{A.17}$$

for $j = 1, \cdots, i-1$ and $k = 1, \cdots, n$. We have the following two conditions.

1) If $k = (j)$ for $j = 1, \cdots, i-1$, then we obtain from (A.17)

$$a_{(j)} = 0 \tag{A.18}$$

or

$$\sum_{r=1}^{n} a_r^2(\lambda_r + \lambda_{(j)}) = -2d_{(j)}^2\lambda_{(j)} + 2\lambda_{(j)}. \tag{A.19}$$

If $d_{(j)} = \pm 1$, then we obtain from (A.19) that all $a_r = 0$ for $r = 1, \cdots, n$. On the other hand, if $d_{(j)} = 0$ then we obtain an equation that is similar to (A.8). We conclude that at most one $a_{(j)} = \pm 1$ and all other $a_r = 0$.

2) If $k \neq (j)$ for $j = 1, \cdots, i-1$, then we obtain an equation that is same as (A.6), from which we conclude that $a_k = 0$, or there is at most one nonzero $a_k = \pm 1$. Thus, the equilibrium points of $J(\boldsymbol{w}_i; A)$ are

$$\boldsymbol{w}_i = d_{(i)}\phi_{(i)} \tag{A.20}$$

where $\phi_{(i)}$ is a permutation of the eigenvectors $\phi_1, \cdots, \phi_n$, and $d_{(i)} = 0, +1$, or -1. Furthermore, for any $j < i$, if $d_{(j)} = 0$, then $\phi_{(i)}$ can be $\phi_{(j)}$.

We can now check that these critical points also satisfy the remaining first order conditions given in (A.3). Be-

sides, equating the remaining terms of (A.3) to zero, does not yield any new stationary point. Thus, all the equilibrium points of the joint objective functions $J(\boldsymbol{w}_i; A)$ for $i = 1, \cdots, p$ are

$$W = [d_{(1)}\phi_{(1)}, \cdots, d_{(i)}\phi_{(i)}, \cdots, d_{(p)}\phi_{(p)}] \tag{A.21}$$

where $d_{(1)}, \cdots, d_{(p)} = 0, +1$, or -1, and $\phi_{(1)}, \cdots, \phi_{(p)}$ are the p permutations of $\phi_1, \cdots, \phi_n$. ■

C. Proof of Theorem 2

From the objective function $J(\boldsymbol{w}_i; A)$ in (12), we define the following energy functions:

$$\begin{aligned} E_i(\boldsymbol{w}_1, \cdots, \boldsymbol{w}_i) =& -2\boldsymbol{w}_i^T A\boldsymbol{w}_i + \boldsymbol{w}_i^T A\boldsymbol{w}_i\boldsymbol{w}_i^T\boldsymbol{w}_i \\ &+ 2\sum_{j=1}^{i-1} \boldsymbol{w}_i^T A\boldsymbol{w}_j\boldsymbol{w}_j^T\boldsymbol{w}_i \end{aligned} \tag{A.22}$$

for $i = 1, \cdots, p$, $(p \le n)$ Let us assume that $d_{(i)} = 0$ for some $i \le p$; i.e.,

$$\begin{aligned} W =& [d_{(1)}\phi_{(1)}, \cdots, d_{(i-1)}\phi_{(i-1)}, 0, \\ & d_{(i+1)}\phi_{(i+1)}, \cdots, d_{(p)}\phi_{(p)}]. \end{aligned}$$

Next, we perturb the ith column of W by $\delta\phi_{(i)}$. We observe that

$$\begin{aligned} & E_i(d_{(1)}\phi_{(1)}, \cdots, d_{(i-1)}\phi_{(i-1)}, \delta\phi_{(i)}) \\ & \quad - E_i(d_{(1)}\phi_{(1)}, \cdots, d_{(i-1)}\phi_{(i-1)}, 0) \\ & = -2\delta^2\lambda_{(i)} + \delta^4\lambda_{(i)}. \end{aligned}$$

Clearly, the energy function decreases for some $\delta < \sqrt{2}$.

We next prove that if $\phi_{(j)} \neq \pm\phi_j$ for some $j = 1, \cdots, p$, then the critical points W are unstable equilibrium points of (A.22). We obtain the Hessian of E_i with respect to $\boldsymbol{w}_i$ as

$$\begin{aligned} & \nabla^2_{\boldsymbol{w}_i} E_i(\boldsymbol{w}_1, \cdots, \boldsymbol{w}_i) \\ & = A\boldsymbol{w}_i^T\boldsymbol{w}_i + 2\boldsymbol{w}_i\boldsymbol{w}_i^T A + \sum_{j=1}^{i-1} A\boldsymbol{w}_j\boldsymbol{w}_j^T + \boldsymbol{w}_i^T A\boldsymbol{w}_i I \\ & \quad + 2A\boldsymbol{w}_i\boldsymbol{w}_i^T + \sum_{j=1}^{i-1} \boldsymbol{w}_j\boldsymbol{w}_j^T A - 2A. \end{aligned} \tag{A.23}$$

We prove the result by induction. We note that

$$\nabla^2_{\boldsymbol{w}_1} E_1(\pm\phi_1) = -A + \lambda_1 I + 4\lambda_1\phi_1\phi_1^T.$$

The eigenvectors of $\nabla^2_{\boldsymbol{w}_1} E_1(\pm\phi_1)$ are $\phi_1, \cdots, \phi_n$. The eigenvalues are $4\lambda_1$ for ϕ_1, and $(\lambda_1 - \lambda_r)$ for ϕ_r $(r > 1)$. Clearly, $\nabla^2_{\boldsymbol{w}_1} E_1(\pm\phi_1)$ is positive definite. On the other hand, we observe that

$$\nabla^2_{\boldsymbol{w}_1} E_1(\pm\phi_r) = -A + \lambda_r I + 4\lambda_r\phi_r\phi_r^T \text{ for } r > 1.$$

The eigenvectors of $\nabla^2_{\boldsymbol{w}_1} E_1(\pm\phi_r)$ are $\phi_1, \cdots, \phi_n$. The eigenvalues are $-(\lambda_1 - \lambda_r)$ for ϕ_1, and $4\lambda_r$ for ϕ_r $(r > 1)$. Clearly, $\nabla^2_{\boldsymbol{w}_1} E_1(\pm\phi_r)$ for $r > 1$ is an indefinite matrix. Thus, $\pm\phi_1$ is a stable equilibrium point of the energy function E_1, whereas $\pm\phi_r$ for $r > 1$ are unstable equilibrium points of E_1.

We next assume that $[d_1\phi_1, \cdots, d_{i-1}\phi_{i-1}]$ $(d_j = \pm 1$ for $j = 1, \cdots, i-1)$ are stable equilibrium points for the energy function E_{i-1}. Then, for $\phi_{(i)} = \pm\phi_i$, we have from (A.23)

$$\begin{aligned} & \nabla^2_{\boldsymbol{w}_i} E_i(d_1\phi_1, \cdots, d_{i-1}\phi_{i-1}, \pm\phi_i) \\ & \quad = -A + \lambda_i I + 4\lambda_i\phi_i\phi_i^T + 2\sum_{j=1}^{i-1} \lambda_j\phi_j\phi_j^T. \end{aligned}$$

The eigenvectors of $\nabla^2_{\boldsymbol{w}_i} E_i(d_1\phi_1, \cdots, d_{i-1}\phi_{i-1}, \pm\phi_i)$ are $\phi_1, \cdots, \phi_n$. The eigenvalues are $(\lambda_i + \lambda_r)$ for ϕ_r $(r < i)$, $4\lambda_i$ for ϕ_i, and $(\lambda_i - \lambda_r)$ for ϕ_r $(r > i)$. Thus, $\nabla^2_{\boldsymbol{w}_i} E_i(d_1\phi_1, \cdots, d_{i-1}\phi_{i-1}, \pm\phi_i)$ is positive definite. On the other hand, if $\phi_{(i)} = \pm\phi_r$ $(r > i)$, then

$$\begin{aligned} & \nabla^2_{\boldsymbol{w}_i} E_i(d_1\phi_1, \cdots, d_{i-1}\phi_{i-1}, \pm\phi_r) \\ & \quad = -A + \lambda_r I + 4\lambda_r\phi_r\phi_r^T + 2\sum_{j=1}^{i-1} \lambda_j\phi_j\phi_j^T. \end{aligned}$$

The eigenvalues of $\nabla^2_{\boldsymbol{w}_i} E_i(d_1\phi_1, \cdots, d_{i-1}\phi_{i-1}, \pm\phi_r)$ are $-(\lambda_i - \lambda_r)$ for ϕ_i, and $4\lambda_r$ for ϕ_r $(r > i)$. Thus, $\nabla^2_{\boldsymbol{w}_i} E_i(d_1\phi_1, \cdots, d_{i-1}\phi_{i-1}, \pm\phi_r)$ is an indefinite matrix. We, therefore, see that $[d_1\phi_1, \cdots, d_i\phi_i]$ $(d_j = \pm 1$ for $j = 1, \cdots, i)$ are the stable equilibrium points of E_i.

From the above analysis, we conclude that $W^* = [\pm\phi_1 \quad \pm\phi_2 \quad \cdots \quad \pm\phi_p]$ are stable equilibrium points of the energy functions (A.22), whereas, $W = [d_{(1)}\phi_{(1)} \ d_{(2)}\phi_{(2)} \ \cdots \ d_{(p)}\phi_{(p)}]$, where $d_{(i)} = 0$ or $\phi_{(i)} \neq \phi_i$ for $i \in \{1, 2, \cdots, p\}$ are unstable equilibrium points.

D. Proof of Theorem 3

Regarding algorithm (21), for simplicity of notation, we use $\boldsymbol{w}_k$ instead of $\boldsymbol{w}_k^1$ for this proof

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \eta_k \left(2A_k\boldsymbol{w}_k - \boldsymbol{w}_k\boldsymbol{w}_k^T A_k\boldsymbol{w}_k - A_k\boldsymbol{w}_k\boldsymbol{w}_k^T\boldsymbol{w}_k\right). \tag{A.24}$$

Let ϕ_k be the first principal eigenvector of A_k corresponding to the largest eigenvalue $\lambda_k = \lambda_{\max}(A_k)$. By multiplying (A.24) on the left by ϕ_k^T, we obtain

$$v_{k+1} = v_k + \eta_k \left(2\lambda v_k - v_k\boldsymbol{w}_k^T A_k\boldsymbol{w}_k - \lambda_k v_k\boldsymbol{w}_k^T\boldsymbol{w}_k\right) \tag{A.25}$$

where $v_k = \phi_k^T\boldsymbol{w}_k$. From (A.25), $v_{k+1}^2 < v_k^2$ if

$$\eta_k < \frac{2}{\boldsymbol{w}_k^T A_k\boldsymbol{w}_k + \lambda_k\boldsymbol{w}_k^T\boldsymbol{w}_k - 2\lambda_k} \ge \frac{1}{\lambda_k(\boldsymbol{w}_k^T\boldsymbol{w}_k - 1)}.$$

Thus, if $\|\boldsymbol{w}_k\|^2 \le \alpha$, then $\|\boldsymbol{w}_{k+1}\|^2 \le \|\boldsymbol{w}_k\|^2$ if

$$\eta_k < \frac{1}{\lambda_{\max}(A_k)(\alpha - 1)}. \tag{A.26}$$

Due to assumption A2, a uniform upper bound for η_k exists. ■

REFERENCES

[1] M. Artin, *Algebra*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[2] P. Baldi and K. Hornik, "Learning in linear neural networks: A survey," *IEEE Trans. Neural Networks*, vol. 6, pp. 837–857, 1995.

[3] S. Bannour and M. R. Azimi-Sadjadi, "Principal component extraction using recursive least squares learning," *IEEE Trans. Neural Networks*, vol. 6, pp. 457–469, 1995.

[4] A. Benveniste, A. Metivier, and P. Priouret, *Adaptive Algorithms and Stochastic Approximations*. New York: Springer-Verlag, 1990.

[5] C. Chatterjee, V. P. Roychowdhury, and E. K. P. Chong, "On relative convergence properties of principal component analysis algorithms," *IEEE Trans. Neural Networks*, vol. 9, pp. 319–329, March 1998.

[6] C. Chatterjee, V. P. Roychowdhury, M. D. Zoltowski, and J. Ramos, "Self-organizing and adaptive algorithms for generalized eigendecomposition," *IEEE Trans. Neural Networks*, vol. 8, pp. 1518–1530, November 1997.

[7] Y. Chauvin, "Principal component analysis by gradient descent on a constrained linear Hebbian cell," in *Proc. Joint Int. Conf. Neural Networks*, vol. I, San Diego, CA, 1989, pp. 373–380.

[8] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*. New York: Wiley, 1993.

[9] Z. Fu and E. M. Dowling, "Conjugate gradient eigenstructure tracking for adaptive spectral estimation," *IEEE Trans. Signal Processing*, vol. 43, pp. 1151–1160, 1995.

[10] ——, "Conjugate gradient projection subspace tracking," in *Proc. 1994 Conf. Signals, Syst. Comput.*, vol. 1, Pacific Grove, CA, 1994, pp. 612–618.

[11] G. H. Golub and C. F. VanLoan, *Matrix Computations*. Baltimore, MD: Johns Hopkins Univ. Press, 1983.

[12] S. Haykin, *Neural Networks—A Comprehensive Foundation*. New York: Macmillan, 1994.

[13] L. Ljung, "Analysis of recursive stochastic algorithms," *IEEE Trans. Automat. Contr.*, vol. AC-22, no. 4, pp. 551–575, August 1977.

[14] D. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1984.

[15] G. Mathew, V. U. Reddy, and S. Dasgupta, "Adaptive estimation of eigensubspace," *IEEE Trans. Signal Processing*, vol. 43, pp. 401–411, 1995.

[16] Y. Miao and Y. Hua, "Fast subspace tracking and neural-network learning by a novel information criterion," *IEEE Trans. Signal Processing*, vol. 46, pp. 1967–1979, 1998.

[17] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *J. Math. Anal. Applicat.*, vol. 106, pp. 69–84, 1985.

[18] M. Plumbley, "Lyapunov function for convergence of principal component algorithms," *Neural Networks*, vol. 8, pp. 11–23, 1995.

[19] T. Okada and S. Tomita, "An optimal orthonormal system for discriminant analysis," *Pattern Recognition*, vol. 18, no. 2, pp. 139–144, 1985.

[20] T. D. Sanger, "Optimal unsupervised learning in a single-layer linear feedforward neural network," *Neural Networks*, vol. 2, pp. 459–473, 1989.

[21] T. K. Sarkar and X. Yang, "Application of the conjugate gradient and steepest descent for computing the eigenvalues of an operator," *Signal Processing*, vol. 17, pp. 31–38, 1989.

[22] L. Xu, "Least mean square error reconstruction principle for self-organizing neural-nets," *Neural Networks*, vol. 6, pp. 627–648, 1993.

[23] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. Signal Processing*, vol. 43, no. 1, pp. 95–107, 1995.

[24] X. Yang, T. K. Sarkar, and E. Arvas, "A survey of conjugate gradient algorithms for solution of extreme eigen-problems of a symmetric matrix," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, pp. 1550–1556, 1989.

[25] W. Zhu and Y. Yang, "Regularized total least squares reconstruction for optical tomographic imaging using conjugate gradient method," in *Proc. Int. Conf. Image Processing*, vol. 1, Santa Barbara, CA, 1997, pp. 192–195.

Chanchal Chatterjee (M'88) received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, and the M.S.E.E. degree from Purdue University, West Lafayette, IN, in 1983 and 1984, respectively. In 1996, he received the Ph.D. degree in electrical and computer engineering from Purdue University.

Between 1985 and 1995, he worked at Machine Vision International and Medar Inc., both in Detroit, MI. He is currently a Senior Algorithms Specialist at BAE Systems, San Diego, CA. He is also affiliated with the Electrical Engineering Department at the University of California, Los Angeles. His areas of interest include image processing, computer vision, neural networks, and adaptive algorithms and systems for pattern recognition and signal processing.

Zhengjiu Kang (M'98) received the B.S. degree in control engineering from Harbin Shipbuilding Engineering Institute, China, in 1991, and the Ph.D. degree in system engineering from Xi'an Jiaotong University, China, in 1996. He is now working toward the Ph.D. degree in wireless communications.

From 1996 to 1997, he was a Researcher at Control Information Systems Lab, Seoul National University, Korea. Since 1997, he has been with the Electrical Engineering Department at University of California, Los Angeles. His current research interests include adaptive algorithms, wireless communications, neural networks, and parallel processing.

Vwani P. Roychowdhury received the B. Tech. degree from the Indian Institute of Technology, Kanpur, India, and the Ph.D. degree from Stanford University, Stanford, CA, in 1982 and 1989, respectively, all in electrical engineering.

From August 1991 until June 1996, he was a Faculty Member at the School of Electrical and Computer Engineering at Purdue University. He is currently a Professor in the Department of Electrical Engineering at the University of California, Los Angeles. His research interests include parallel algorithms and architectures, design and analysis of neural networks, application of computational principles to nanoelectronics, special purpose computing arrays, VLSI design, and fault-tolerant computation. He has coauthored several books, including *Discrete Neural Computation: A Theoretical Foundation* (Englewood Cliffs, NJ: Prentice-Hall, 1995) and *Theoretical Advances in Neural Computation and Learning* (Boston, MA: Kluwer, 1994).