Available at www.ComputerScienceWeb.com
POWERED BY SCIENCE DIRECT®

Signal Processing 83 (2003) 307–324

SIGNAL PROCESSING

www.elsevier.com/locate/sigpro

Fast adaptive eigenvalue decomposition: a maximum likelihood approach

Thierry Chonavel[a,*], Benoît Champagne[b], Christian Riou[a]

[a] ENST Bretagne, Dept. SC, BP 832, 29286 Brest cedex, France
[b] McGill University, 3480 University St., Montreal, Que., Canada H3A 2A7

Received 24 January 2000; received in revised form 9 August 2002

Abstract

In this paper, we address the problem of adaptive eigenvalue decomposition (EVD). We propose a new approach, based on the optimization of the log-likelihood criterion. The parameters of the log-likelihood to be estimated are the eigenvectors and the eigenvalues of the data covariance matrix. They are actualized by means of a stochastic algorithm that requires little computational cost. Furthermore, the particular structure of the algorithm, that we named MALASE, ensures the orthonormality of the estimated basis of eigenvectors at each step of the algorithm. MALASE algorithm shows strong links with many Givens rotation based update algorithms that we discuss. We consider convergence issues for MALASE algorithm and give the expression of the asymptotic covariance matrix of the estimated parameters. The practical interest of the proposed method is shown on examples.

Keywords: Maximum likelihood; Eigenvalue decomposition; Subspace tracking; Unitary matrix; Matrix manifold; Givens rotation

1. Introduction

Let $X = (X_n)_{n\in\mathbb{Z}}$ denote a size-N, complex valued process ($X_n \in \mathbb{C}^N$), with covariance matrix $R_{X,n} = E\{X_n X_n^{\mathrm{H}}\}$. Subspace methods are based on the eigenvalue decomposition (EVD) of $R_{X,n}$, or at least on the estimation of the subspace spanned by the eigenvectors associated to the largest (resp. the smallest) eigenvalues, and named the signal (resp. the noise) subspace. These methods are classically used within several multivariate signal processing problems.

Applications such as sources localization (MUSIC algorithm, [5,25]), vector quantization [15], or more recently blind channel equalization impulse response identification [19], often require estimation of such decompositions of the covariance matrices.

When X is a wide sense stationary process, the covariance matrix $R_{X,n}$ is constant ($R_{X,n} = R_X$), and it can be estimated easily by means of any classical technique, such as the empiric estimator (i.e. in the form $(1/L)\sum_{k=1,L} X_k X_k^{\mathrm{H}}$, where L is the data length). Then, the corresponding eigenvalue decomposition can be obtained from standard algorithms such as the orthogonal iteration method (e.g. [16]).

But, in many applications, such as moving source localization and tracking, or time varying channel

* Corresponding author.
E-mail addresses: thierry.chonavel@enst-bretagne.fr (T. Chonavel), champagne@ece.mcgill.ca (B. Champagne).

0165-1684/03/$ - see front matter © 2002 Elsevier Science B.V. All rights reserved.
PII: S0165-1684(02)00417-6

identification for mobile radio transmissions, the covariance matrix $R_{X,n}$ is time varying and the above methods no longer apply directly. In this case, several approaches have been proposed to track the instantaneous EVD of $R_{X,n}$ (see [10] for an excellent survey of adaptive subspace estimation prior to 1990; some more recent algorithms are reviewed in [12]).

For instance, a first kind of techniques consists in adapting some classical EVD computation algorithm for adaptive purposes via an extension from the stationary case to the non-stationary case. Generally, the covariance matrix estimate is computed by

$$\hat{R}_{X,n} = (1-\mu)\hat{R}_{X,n-1} + \mu \mathbf{x}_n \mathbf{x}_n^{\mathrm{H}}, \tag{1}$$

where $0 < \mu \ll 1$, and $\mathbf{x}_n$ represents the data vector at time n, i.e. the realization of the process X at time n. Then, the EVD of $\hat{R}_{X,n}$ is performed; in fact, at each instant n, only one (or a few) iteration of a classical EVD algorithm is performed to update the previously estimated EVD. This iteration may also involve an orthonormalization procedure that requires a QR or GS decomposition. Equivalent approaches have also been proposed to perform adaptive singular value decompositions (SVD) of the data matrix. Algorithms of these kinds can be found for example in [14,18,20,26], with computational complexity ranging from $O(N^2)$ or $O(NK^2)$ to $O(NK)$, where K is the dimension of the subspace to be estimated.

In some approaches the estimate of $R_{X,n}$ is directly replaced in Eq. (1) by the parameters of its EVD. Then, the subspaces are updated by techniques that involve Householder reflection matrices, or Givens rotation matrices (see e.g. [11] and [9], respectively). With such parametrizations of the eigenvectors update, the orthonormality of the estimated basis is held at each iteration. Let us remark that in [11] the signal and noise subspaces eigenvalues are averaged. Thus, only the estimation of a basis of both subspaces can be obtained. On the contrary, in [9] the complete EVD can be estimated. Both methods require $O(NK)$ operations.

Another kind of algorithms is designed to estimate the EVD (or a basis of the subspace of interest) by performing adaptive optimization of some possibly constrained criterion. This is done by means of some stochastic gradient algorithm (e.g. [21–23]), or as in [1,28,29] by means of an RLS approach. The computational cost of these methods ranges from $O(NK)$ to $O(NK^2)$, or $O(N^2K)$.

Some of the above methods simply intend to provide a basis of the signal or of the noise subspace, while some others supply an estimate of the complete EVD. Many applications make use of the projector onto the noise or signal subspace, and as pointed out in [9] perfect orthonormality is a desirable property for many subspace-based estimation techniques. In particular, for sources localization, deviations from the orthonormality lead to limited resolution capability and limited estimation performance of the root MUSIC algorithm [3]. Here, our main interest for considering the orthonormality issue stems from the fact that stochastic algorithms may sometimes become unstable if some constraints upon the parameters are not satisfied. In particular, we will see that we must ensure an orthonormality constraint upon the eigenbasis to derive a stable adaptive EVD based on the log-likelihood criterion.

Besides, it is often interesting to get the complete EVD, involving in particular the estimation of the eigenvalues, to solve problems where the noise or signal subspace dimension has to be estimated. Indeed, the knowledge of the eigenvalues allows the use of statistical criteria to estimate their dimension (e.g. [2,24]).

In this paper, we propose a new approach to adaptively estimate the EVD, that provides a number of desirable features, as explained below. Unlike the above mentioned techniques, it is based on adaptive optimization of the log-likelihood functional. This is quite a natural criterion for statistical estimation purposes, even if the minimum variance property of the log-likelihood functional is actually an asymptotic property. With this criterion, we could derive an algorithm that provides exact orthonormality of the estimated eigenvectors, with computational complexity in $O(N^2)$ for the calculation of the complete EVD.

This algorithm is closely related to algorithms such as PROTEUS-1 and PROTEUS-2 [9]. In particular, using the same preprocessing technique as in [9], it is possible to reduce the computational burden: when only $K < N$ eigenvalues of $R_{X,n}$ are distinct (as in many subspace tracking problems),

the update of the eigendecomposition can be performed at the expense of $O(NK)$ operations. Furthermore, we have shown that the only stable stationary points of the new algorithm represent the EVD of the data covariance matrix. Also, we calculated the asymptotic covariance matrix for the parameters of the EVD. For the sake of conciseness, the calculations related to convergence and asymptotic covariance are omitted here, and will be published separately.

The paper is organized as follows: Section 2 deals with the derivation of the algorithm, that we named MAximum Likelihood Adaptive Subspace Estimation (MALASE). Reduction of the computational burden and links with algorithms based on updates that involve Givens rotations are considered in Section 3, as well as the convergence and asymptotic performance. Section 4 is devoted to examples that show the satisfactory behavior of the algorithm. Some concluding remarks are given in Section 5.

To end this section, let us introduce some notations. First, the superscript $^{\mathrm{T}}$, *, and $^{\mathrm{H}}$ will respectively denote the transpose, conjugate, and transpose–conjugate of a vector or a matrix. The trace, determinant, and expectation operators will be denoted by $\mathrm{Tr}(.)$, $|.|$, and $E\{.\}$ respectively. Vectors will be denoted by bold lower case letters, as in $\mathbf{v}$, with the corresponding ith entry denoted as v_i. In addition, $\mathbf{v} = \mathrm{diag}(A)$ will denote the vector $\mathbf{v}$ that contains the diagonal terms of the matrix A, while $A = \mathrm{diag}(v)$ will denote the diagonal matrix with diagonal terms $[A]_{i,i} = v_i$ ($[A]_{i,j}$ denotes the ijth entry of A). I_K will denote the size K identity matrix. In general, the (estimated) EVD of $R_{X,n}$ will be denoted $U_n^{\mathrm{H}} \Lambda_n U_n$, where U_n is the unitary matrix that contains the eigenvectors, denoted $(U_{i,n})_{i=1,N}$ ($U_n = [U_{1,n}, \ldots, U_{N,n}]$), and $\Lambda_n = \mathrm{diag}(\lambda_{1,n}, \ldots, \lambda_{N,n})$ is the diagonal matrix that contains the eigenvalues $(\lambda_{i,n})_{i=1,N}$ at time n. When necessary, we will assume without loss of generality that the eigenvalues are classified in decreasing order.

2. Derivation of MALASE algorithm

In this section, we derive the proposed algorithm. It is based on iterative optimization of the log-likelihood functional, where the parameters to be estimated are those of the EVD of $R_{X,n}$. In particular, we show how it is possible to account for an orthonormality constraint upon U_n at each step of the algorithm.

2.1. The maximum-likelihood criterion

Let f denote the probability density function of the random, complex circular, and zero mean Gaussian vector, denoted X_n, with $X_n \in \mathbb{C}^N$. Under the hypothesis $R_{X,n} = R$, we have

$$f(\mathbf{x}_n; R) = \frac{1}{\pi^N |R|} \exp(-\mathbf{x}_n^{\mathrm{H}} R^{-1} \mathbf{x}_n). \tag{2}$$

It is clear that up to an additive constant, the opposite of the log-likelihood is given by

$$\Psi(\mathbf{x}_n; R) = \log |R| + \mathbf{x}_n^{\mathrm{H}} R^{-1} \mathbf{x}_n. \tag{3}$$

Letting $U \Lambda U^{\mathrm{H}}$ represent the EVD of R, with $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$, $\Psi(\mathbf{x}_n, R)$ can be rewritten in the form

$$\Phi(\mathbf{x}_n; U, \Lambda) = \sum_{k=1,N} \log \lambda_k + \mathbf{x}_n^{\mathrm{H}} U \Lambda^{-1} U^{\mathrm{H}} \mathbf{x}_n. \tag{4}$$

It might seem natural to iteratively search a minimum of Φ by means of a gradient algorithm.

It is shown in Appendix A that a direct update of (U, Λ) by means of a stochastic gradient approach would not lead to a converging algorithm. In what follows, we are going to show that it is possible to overcome this problem by accounting for a unitary constraint upon U.

2.2. The proposed updating scheme

In [13] matrix based algorithms are studied from the viewpoint of continuous-time systems theory and differential geometry. With a view to optimize $\Phi(\mathbf{x}_n; U, \Lambda)$ let us recall the following result:

Proposition 1. *Let $U(t)$ be a differentiable matrix function defined on the manifold of unitary matrices. Then, the gradient vector of the function $\mathrm{Tr}[(UH_1U^{\mathrm{H}})H_2]$, where H_1 and H_2 are arbitrary hermitian matrices, is given by*

$$\mathrm{grad}_U \mathrm{Tr}[(UH_1U^{\mathrm{H}})H_2] = [H_2(UH_1U^{\mathrm{H}}) - (UH_1U^{\mathrm{H}})H_2]U, \tag{5}$$

where $\mathrm{grad}_U f(U)$ *represents the matrix with* (i,j)*th entry in the form* $\partial f(U)/\partial U_{ij}$.

Proof (see [13, p. 62–63]).

Since $\Phi(\mathbf{x}_n; U, \Lambda)$ can be written as

$$\Phi(\mathbf{x}_n; U, \Lambda) = \sum_{k=1,N} \log \lambda_k + \mathrm{Tr}[(U\Lambda^{-1}U^{\mathrm{H}})(\mathbf{x}_n\mathbf{x}_n^{\mathrm{H}})],$$

it comes from (5) that

$$\begin{aligned}\mathrm{grad}_U \Phi(\mathbf{x}_n; U, \Lambda) &= [\mathbf{x}_n\mathbf{x}_n^{\mathrm{H}}(U\Lambda^{-1}U^{\mathrm{H}}) \\ &\quad - (U\Lambda^{-1}U^{\mathrm{H}})\mathbf{x}_n\mathbf{x}_n^{\mathrm{H}}]U \\ &= U(\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda^{-1} - \Lambda^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}), \qquad (6)\end{aligned}$$

where $\mathbf{y}_n = U^{\mathrm{H}}\mathbf{x}_n$. Furthermore, recalling that the exponential of a square matrix A is defined by $\exp(A) = \sum_{k=0}^{\infty} A^k/k!$, it is clear that $\mathrm{grad}_U \Phi(\mathbf{x}_n; U, \Lambda)$ is tangent to the curve defined by

$$U(t) = U \exp[t(\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda^{-1} - \Lambda^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}})]$$

for $t = 0$. The reader will note that this curve lies in the manifold of unitary matrices. Indeed, a square matrix, say A, is a skew-symmetric matrix (i.e. $A^{\mathrm{H}} = -A$) if and only if $\exp(A)$ is a unitary matrix (see e.g. [9,27]), and matrix $\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda_{n-1}^{-1} - \Lambda_{n-1}^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}$ is clearly skew-symmetric.

Moving on the curve $U(t)$ from point $t = 0$ in the direction of decreasing values of $\Phi(\mathbf{x}_n; U, \Lambda)$ amounts to let t decrease. Thus, a discretized version of the optimization of $\Phi(\mathbf{x}_n; U, \Lambda)$ as a continuous function of U is given by the following update scheme:

$$U_n = U_{n-1} \exp(\mu(\Lambda_{n-1}^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda_{n-1}^{-1})), \qquad (7)$$

where μ is the algorithm stepsize. Similarly Λ can be updated by means of a gradient technique, yielding the following update algorithm for (U_n, Λ_n):

$$\mathbf{y}_n = U_{n-1}^{\mathrm{H}}\mathbf{x}_n, \qquad (8a)$$

$$U_n = U_{n-1} \exp(\mu(\Lambda_{n-1}^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda_{n-1}^{-1})), \qquad (8b)$$

$$\Lambda_n = \Lambda_{n-1} + \mu'(\Lambda_{n-1}^{-2}\,\mathrm{diag}(\mathrm{diag}(\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}})) - \Lambda_{n-1}^{-1}). \qquad (8c)$$

We will call this algorithm as MALASE, for MAximum Likelihood Adaptive Subspace Estimation. The stepsizes μ and μ' in (8) are possibly different. The reason for choosing different stepsizes lies in the fact that the convergence rate may be different for U_n and Λ_n when $\mu = \mu'$, as accounted for in the simulations of Section 4.

For those readers who may not be familiar with derivation on matrix manifolds, an alternative direct derivation of MALASE is supplied in the Appendix B.

2.3. Some properties of MALASE algorithm

Let us briefly point out some nice features of MALASE algorithm.

2.3.1. Orthonormality

As mentioned above the update factor $\exp(\mu \times(\Lambda_{n-1}^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda_{n-1}^{-1}))$ is a unitary matrix. This ensures that the orthonormality property is preserved by MALASE algorithm, provided that the algorithm is initialized with a unitary matrix U_0. However, it is not necessary to have U_0 unitary to ensure the convergence since MALASE algorithm steers the matrix U towards the manifold of unitary matrices (see the simulation part).

2.3.2. Computational complexity

MALASE algorithm may seem to involve high computational cost, due to the matrix exponential that appears in relation (8b). However, since $\exp(\mu(\Lambda_{n-1}^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda_{n-1}^{-1})$ is the exponential of the sum of two rank one matrices, it is possible to show that the calculation of this matrix requires only $O(N^2)$ operations. To check this, let us denote $\mathbf{z}_n = \Lambda_{n-1}^{-1}\mathbf{y}_n$, and $[\mathbf{z}_n\,\mathbf{y}_n] = Q_nR_n$ the QR decomposition of the size-$N \times 2$ matrix $[\mathbf{z}_n\,\mathbf{y}_n]$, with Q_n of size $N \times 2$. Then, letting $c_n = \sqrt{||\mathbf{y}_n||^2||\mathbf{z}_n||^2 - (\mathbf{y}_n^{\mathrm{H}}\mathbf{z}_n)^2}$ it can be shown (see Appendix C) that

$$\begin{aligned}&\exp(\mu(\Lambda_{n-1}^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda_{n-1}^{-1})) \\ &\quad = I + Q_n \begin{pmatrix} \cos\mu c_n - 1 & -\sin\mu c_n \\ \sin\mu c_n & \cos\mu c_n - 1 \end{pmatrix} Q_n^{\mathrm{H}}. \qquad (9)\end{aligned}$$

It is easy to check that the calculation of $U_{n-1}\exp(\mu \times(\Lambda_{n-1}^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda_{n-1}^{-1}))$ only requires $O(N^2)$

operations. We will explain in Section 3 how this computational burden can be decreased when $R_{X,n}$ has less than N distinct eigenvalues. Finally, MALASE algorithm can be implemented as follows:

MALASE algorithm

Initialize U_0 unitary,
$\lambda_{k,0}$ with distinct values $(k = 1, N)$,
and $\mu, \mu' > 0$,

For each time step do

input $\mathbf{x}_n$

$$\mathbf{y}_n = U_{n-1}^{\mathrm{H}} \mathbf{x}_n$$

$$z_{k,n} = \lambda_{k,n-1}^{-1} y_{k,n} \qquad (k = 1, N)$$

$$c_n = \sqrt{||\mathbf{y}_n||^2 ||\mathbf{z}_n||^2 - (\mathbf{y}_n^{\mathrm{H}} \mathbf{z}_n)^2}$$

$$Q_n = \left[\frac{\mathbf{z}_n}{\|\mathbf{z}_n\|} \; \frac{\mathbf{z}_n(\mathbf{z}_n^{\mathrm{H}} \mathbf{y}_n) - \mathbf{y}_n \|\mathbf{z}_n\|^2}{c_n \|\mathbf{z}_n\|} \right] \tag{10}$$

$$U_n = U_{n-1} + \{U_{n-1} Q_n\} \times \left\{ \begin{pmatrix} \cos \mu c_n - 1 & -\sin \mu c_n \\ \sin \mu c_n & \cos \mu c_n - 1 \end{pmatrix} Q_n^{\mathrm{H}} \right\}$$

$$\lambda_{k,n} = \lambda_{k,n-1} + \mu' (\lambda_{k,n-1}^{-2} |y_{k,n}|^2 - \lambda_{k,n-1}^{-1}) \qquad (k = 1, N).$$

The $\lambda_{k,0}$ can be chosen for instance regularly spaced in the interval $[a, b]$, where a and b are rough estimates of the smallest eigenvalue and of the largest eigenvalue. U_0 can be chosen equal to the identity matrix. Also, it is possible to perform an initial EVD, SVD, QR or GS decomposition from the first data samples in order to get a better initialization.

3. Reduction of the computational burden and convergence properties

3.1. Reduction of the computational burden

Now, let us assume that we are only interested in searching for the subspace associated to the p largest, or the p smallest, eigenvalues of $R_{X,n}$. We are going to show that in this case MALASE algorithm can be implemented in a way that requires updating only $p + 1 = K$ vectors at each iteration, and that this can be done at the expense of $O(NK)$ operations. This $O(NK)$ implementation will be referred to as **MALASE**(K).

We assume that the $N - p$ eigenvalues associated with the eigenvectors that we do not need are all equal. This situation arises in many applications. When this condition is not matched, it is possible to average these $N - p$ eigenvalues, as proposed in [11].

Let us assume for instance that we search for vectors $U_{1,n}, \ldots, U_{p,n}$ and that the $N - p$ smallest eigenvalues are all equal: $\lambda_{p+1} = \cdots = \lambda_N$. As in [9], we achieve computational complexity reduction by means of a convenient preprocessing of the vector $\mathbf{x}_n$, that replaces the calculation of $\mathbf{y}_n = U_n^{\mathrm{H}} \mathbf{x}_n$.

The preprocessing is based on an isometric transform applied to $\mathbf{x}_n$. $\mathbf{x}_n$ is transformed into a size $p+1$ vector, denoted ξ_n, with first p components representing the projection of $\mathbf{x}_n$ on the space spanned by $U_{1,n}, \ldots, U_{p,n}$. Then, all but the first component of the projection of $\mathbf{x}_n$ on the space spanned by $U_{p+1,n}, \ldots, U_{N,n}$ are set to zero by means of a bloc-Householder transform, denoted H_n (see e.g. [16] for more details). The non-zero component defines $\xi_{p+1,n}$. These operations amount to calculate $H_n \mathbf{y}_n = [\xi_n^{\mathrm{T}} 0^{\mathrm{T}}]^{\mathrm{T}}$, but we are going to see that this is done in a way that requires only $O(NK)$ operations. The following summary of the preprocessing shows this clearly.

Preprocessing

For each time step do

input $\mathbf{x}_n$

$$U_S = [U_{1,n-1}, \ldots, U_{p,n-1}]$$

$$a = U_S^{\mathrm{H}} \mathbf{x}_n \tag{11}$$

$$b = \mathbf{x}_n - U_S a$$

$$\xi_n = [a^{\mathrm{T}} \; \|b\|]^{\mathrm{T}}$$

Let us remark that unlike in [9] no preprocessing in the form $a \to Da$, where $D=\mathrm{diag}(a_1^*/|a_1|,\ldots,a_p^*/|a_p|)$ is required.

Now, let us denote $v_n=\mathrm{diag}(\lambda_{1,n-1}^{-1},\ldots,\lambda_{p+1,n-1}^{-1})\xi_n$. Let us remark that $H_n\mathbf{z}_n=[v_n^{\mathrm{T}}0^{\mathrm{T}}]^{\mathrm{T}}$ since $\lambda_{p+1,n-1}=\cdots=\lambda_{N,n-1}$. It is clear that the terms c_n and Q_n in MALASE algorithm can be rewritten as

$$c_n=\sqrt{||\xi_n||^2||v_n||^2-(\xi_n^{\mathrm{H}}v_n)^2},$$

$$Q_n=H_n^{\mathrm{H}}\begin{pmatrix}Q_n^{(K)}\\0\end{pmatrix}, \tag{12}$$

$$Q_n^{(K)}=\left[\frac{v_n}{\|v_n\|}\ \frac{v_n(v_n^{\mathrm{H}}\xi_n)-\xi_n\|v_n\|^2}{c_n\|v_n\|}\right].$$

Then, the update of U_n can be rewritten as

$$U_n=U_{n-1}+U_{n-1}H_n^{\mathrm{H}}\times\begin{pmatrix}Q_n^{(K)}\begin{pmatrix}\cos\mu c_n-1 & -\sin\mu c_n\\ \sin\mu c_n & \cos\mu c_n-1\end{pmatrix}Q_n^{(K)\mathrm{H}} & 0\\ 0 & 0\end{pmatrix}H_n, \tag{13}$$

i.e.

$$U_nH_n^{\mathrm{H}}=U_{n-1}H_n^{\mathrm{H}}+U_{n-1}H_n^{\mathrm{H}}\times\begin{pmatrix}Q_n^{(K)}\begin{pmatrix}\cos\mu c_n-1 & -\sin\mu c_n\\ \sin\mu c_n & \cos\mu c_n-1\end{pmatrix}Q_n^{(K)\mathrm{H}} & 0\\ 0 & 0\end{pmatrix}. \tag{14}$$

Let us denote $U_n^{(K)}$ the $N\times(p+1)$ matrix that contains the first $p+1$ vectors of $U_nH_n^{\mathrm{H}}$. It is clear that the first p columns of U_n, are left unchanged by the transform $U\to UH_n^{\mathrm{H}}$. The last column of $U_n^{(K)}$ is $U_{p+1,n}^{(K)}=\|b\|^{-1}b$, where b is defined as in the preprocessing. Thus, $U_n^{(K)}=[U_{1,n},\ldots,U_{p,n},\|b\|^{-1}b]$.

From these considerations, it appears that MALASE algorithm can be rewritten in a new way that we will name MALASE(K).

MALASE(K) implementation of MALASE algorithm

Initialize U_0 unitary, initialize $\lambda_{k,0}$ with distinct values $(k=1,K)$, and $\mu,\mu'>0$,

For each time step do

input $\mathbf{x}_n$

$$U_S=[U_{1,n-1}^{(K)},\ldots,U_{p,n-1}^{(K)}]$$
$$a=U_S^{\mathrm{H}}\mathbf{x}_n$$
$$b=\mathbf{x}_n-U_Sa$$
$$\xi_n=[a^{\mathrm{T}}\ \|b\|]^{\mathrm{T}}$$
$$U_{p+1,n-1}^{(K)}=\|b\|^{-1}b$$
$$v_{k,n}=\lambda_{k,n-1}^{-1}\xi_{k,n},\quad (k=1,K)$$
$$c_n=\sqrt{||\xi_n||^2||v_n||^2-(\xi_n^{\mathrm{H}}v_n)^2} \tag{15}$$
$$Q_n^{(K)}=\left[\frac{v_n}{\|v_n\|}\ \frac{v_n(v_n^{\mathrm{H}}\xi_n)-\xi_n\|v_n\|^2}{c_n\|v_n\|}\right]$$
$$U_n^{(K)}=u_{n-1}^{(K)}+\{U_{n-1}^{(K)}Q_n^{(K)}\}\times\left\{\begin{pmatrix}\cos\mu c_n-1 & -\sin\mu c_n\\ \sin\mu c_n & \cos\mu c_n-1\end{pmatrix}Q_n^{(K)\mathrm{H}}\right\}$$
$$\lambda_{k,n}=\lambda_{k,n-1}+\mu'(\lambda_{k,n-1}^{-2}|\xi_{k,n}|^2-\lambda_{k,n}^{-1})\quad(k=1,K-1)$$
$$\lambda_{K,n}=\lambda_{K,n-1}+\mu'(\lambda_{K,n-1}^{-2}\frac{|\xi_{K,n}|^2}{N-p}-\lambda_{K,n}^{-1}).$$

Since $Q_n^{(K)}$ is a size $K\times 2$ matrix, it is clear that MALASE(K) requires only $O(NK)$ operations. Some important issues for practical implementation of MALASE(K) are presented in Sections 4.2.1 and 4.2.2.

3.2. EVD update and Givens rotations

Several eigenvalue update techniques, such as MALASE or PROTEUS-1 algorithm [9], rely on an

update of U_n in the form

$$U_n = U_{n-1}\exp(\Theta_n), \tag{16}$$

where Θ_n is a skew symmetric matrix, or in the form

$$U_n = U_{n-1}\Pi_{1\leqslant i<j\leqslant N} G_{i,j}(\theta_{ij,n}), \tag{17}$$

where $G_{i,j}(\theta_{ij,n})$ is a Given's rotation that operates on columns i and j (see e.g. [13] and enclosed references). In general both updates (16) and (17) are equivalent up to a second order term in $O(\mu^2)$, letting $\theta_{ij,n} = [\Theta_n]_{ij}$. In particular, MALASE yields (for the real case)

$$\theta_{ij,n} = \mu\left(\frac{1}{\lambda_{i,n-1}} - \frac{1}{\lambda_{j,n-1}}\right)\xi_{i,n}\xi_{j,n}. \tag{18}$$

In [9], PROTEUS-1 and PROTEUS-2 algorithms involve rotations in the form

$$\theta_{ij,n} = \frac{\mu}{\lambda_{j,n-1} - \lambda_{i,n-1}}\,\xi_{i,n}\xi_{j,n},$$

and

$$\theta_{ij,n} = -\frac{\mu}{\lambda_{i,n-1}}\,\xi_{i,n}\xi_{j,n} \quad \text{for } i<j,$$

$$\theta_{ji,n} = -\theta_{ij,n}, \tag{19}$$

yielding computational costs in $O(NK^2)$ and $O(NK)$ operations per step respectively.

It is interesting to note that if $\forall i<j$, $\lambda_i \gg \lambda_j$, then for MALASE $\theta_{ij,n}$ can be approximated by $\theta_{ij,n} = -(\mu/\lambda_{j,n-1})\xi_{i,n}\xi_{j,n}$. Then $|\theta_{ij,n}|$ is larger for MALASE than for PROTEUS-2. This is in accordance with the faster convergence and higher asymptotic variance often observed for MALASE compared to PROTEUS-2 when μ is fixed.

Note also that PROTEUS algorithms are derived from the following update of $R_{X,n}$:

$$\hat{R}_{X,n} = (1-\mu)\hat{R}_{X,n-1} + \mu\mathbf{x}_n\mathbf{x}_n^{\mathrm{H}}. \tag{20}$$

From this, and above mentioned similarities between MALASE and PROTEUS algorithms, we get an interpretation of the update factor μ of MALASE algorithm: μ can be seen as the update factor of the stochastic gradient estimate of $R_{X,n}$.

3.3. Convergence and asymptotic covariance matrix

As usual, the study of the convergence is considered for a stationary random process X. It has been possible to prove that stationary stable points of the algorithm correspond to the desired solution and to calculate the asymptotic variance of the parameters. However, the convergence analysis is rather involved and beyond the scope of this paper. For this reason we hope to present it separately, and only the main results are collected in this section.

3.3.1. Convergence of MALASE algorithm

Let (U_*, Λ_*) represent a stationary point of MALASE algorithm. Then rewriting equations (8) in the form

$$\begin{aligned} U_{n+1} &= U_n + \mu H_1(X_n; \Lambda, U),\\ \Lambda_{n+1} &= \Lambda_n + \mu H_2(X_n; \Lambda, U), \end{aligned} \tag{21}$$

it can be shown that the stationary points correspond to unitary matrices U such that $U^{\mathrm{H}}R_X U$ is a block diagonal matrix (the sizes of the different blocks may be different), where the terms along the diagonal of a given block are all equal. Thus it appears that the eigenvalue decompositions are not the only stationary points of the algorithm. However, the following result ensures the desired convergence property:

Proposition 2. *The stationary stable points of MALASE algorithm correspond to the EVD of R_X.*

3.3.2. Asymptotic covariance matrix

Using standard techniques involving linearization of the update of U_n for small μ and results of the ordinary differential equation theory [4], we have been able to derive the asymptotic covariance matrix for the parameters.

Let $(U_{1,n},\ldots,U_{N,n})$ and $(\lambda_{1,n},\ldots,\lambda_{N,n})$ denote the estimated eigenvectors and the estimated eigenvalues, and let us denote $V_n = [U_{1,n}^{\mathrm{T}},\ldots,U_{N,n}^{\mathrm{T}},\lambda_{1,n},\ldots,\lambda_{N,n}]^{\mathrm{T}}$ the size-(N^2+N) vector that contains these parameters. For small values of μ the asymptotic covariance matrix of V_n is equal to μP, where P is the only solution of a Lyapunov equation; this solution is in the form

$$\mu P = \frac{1}{2}\,\mu I_{N^2+N}. \tag{22}$$

This is a very interesting result since the asymptotic variance depends only on μ, and not on the parameters of the EVD of R_X, unlike other algorithms. Furthermore, any of the N^2+N parameters are asymptotically uncorrelated, and have the same variance, equal to $\mu/2$. For eigenvalues, this can be compared for instance to the usual update in the form

$$\lambda_{k,n} = \lambda_{k,n-1} + \mu(|\xi_{k,n}|^2 - \lambda_{k,n-1}), \tag{23}$$

that yields theoretical asymptotic variance for λ_k equal to $(\mu/2)\lambda_k^2$.

3.3.3. Convergence rate

Paralleling the discussion in [7] where it is shown that PROTEUS-2 has similar convergence rates for all the parameters, it can be shown that this property may not be satisfied for MALASE, as already mentioned in Section 2.2. Using (23) for the update of the eigenvalues in MALASE would permit to change this. But then, the asymptotic variance of MALASE would no longer be similar for all the parameters.

4. Experimental results

4.1. Asymptotic variance

The above considerations upon the asymptotic covariance of the algorithm are derived for small values of μ. But in practice, reasonable values of μ, i.e. values of μ that ensure convergence at a speed about equivalent to that of other existing techniques, cannot be considered as small for MALASE algorithm. In particular, with X defined as in Section 4.2, we calculated the average variance of the estimated eigenvectors coefficients from 10^4 iterations of the algorithm. We obtained 6×10^{-5} for $\mu=10^{-4}$, 0.0042 for $\mu=0.005$, and 0.072 for $\mu=0.05$. Thus, only for $\mu=10^{-4}$ the variance of the parameters is about $\mu/2$. This shows that for large values of μ, one should better consider values of the asymptotic covariance calculated from Monte Carlo simulations. However, the theoretical asymptotic variance is a useful index since to a certain extent the simulation results confirm the theoretical results. In accordance with theoretical variance calculations, we have checked that the asymptotic variance of distinct parameters do not differ too much for MALASE, especially for low SNR (i.e. when the eigenvalues are little scattered). On the contrary, for algorithms that use update (23), the variance of the estimated eigenvalues increases for increasing eigenvalues.

4.2. Convergence of MALASE algorithm

Now, let us consider the following example proposed in [9]. We assume that

$$X_n = \sum_{k=1,p} S_{k,n} d(\omega_k) + W_n, \tag{24}$$

where $d(\omega) = [1, e^{i\omega}, \ldots, e^{i(N-1)\omega}]^T$. $(S_{k,n})_{n\in\mathbb{Z}}$ is a complex scalar, white circular Gaussian source signal, and $(W_n)_{n\in\mathbb{Z}}$ a white Gaussian noise vector with covariance matrix $\sigma_W^2 I_N$. The noise and the sources are uncorrelated. All sources have the same power, and the signal to noise ratio (SNR) is defined as $E\{|S_{k,n}|^2\}/\sigma_W^2$.

We study the convergence of MALASE algorithm, implemented in the form MALASE(K), with $K = p+1$. To this end, we consider the performance index defined by

$$\mathrm{err}_n = \|P_S - U_{S,n}U_{S,n}^H\|, \tag{25}$$

where P_S is the true projector on the source subspace (i.e. on the subspace spanned by the set $\{d(\omega_1),\ldots,d(\omega_p)\}$), $U_{S,n}U_{S,n}^H$ the estimated projector, and $\|A\| = [\mathrm{Tr}(AA^H)]^{1/2}$. In other words, $U_{S,n}$ is an $N\times p$ matrix, and its columns are the eigenvectors associated with the p largest eigenvalues of R_X, estimated at time n.

We take $N=16$ and $p=4$. The values of the $(\omega_k)_{k=1,p}$ are first set to 0.0, 0.25, 1, and 1.25. We study the results obtained for MALASE, and we compare them to the $O(NK)$ algorithms PROTEUS-2 and LORAF-3 [26]. Also we plot the evolution of err_n for Karasalo's method [17]. This method shows high computational cost (in $O(NK^2)$), but its convergence and tracking behaviors are excellent, and thus is interesting as a reference.

In order to compare the convergence speeds of the algorithms, their stepsizes are adapted so as to ensure the same value of the asymptotic floor for the performance index. Table 1 gives the values for the stepsize for the different algorithms that have been chosen to ensure a floor at 0.2 for SNR $=-5$ dB and 0.05 for SNR $=10$ dB, respectively.

Table 1
Stepsizes for simulations where $\omega = [0, 0.25, 1, 1.25]$

	SNR	
	−5 dB	10 dB
Karasalo	$\mu = 0.0075$	$\mu = 0.02$
MALASE	$\mu = 0.0035$ $\mu' = 0.35$ $\theta_{\min} = 0.02$ $\beta = 1$	$\mu = 0.00018$ $\mu' = 100$ $\theta_{\min} = 0.01$ $\beta = 10$
PROTEUS-2	$\mu = 0.01$	$\mu = 0.0165$
LORAF-3	$\alpha = 1 - 0.002$	$\alpha = 1 - 0.00455$

The convergence is studied at low SNR (−5 dB) and at high SNR (10 dB). The eigenbasis is initialized with identity matrix and the eigenvalues are initialized from a rough estimate of the largest eigenvalue: we choose them equal to $(10/(p+1)) \times [p+1,\ p, \ldots, 1]$ for SNR = −5 dB, and $(200/(p+1)) \times [p+1, p, \ldots, 1]$ for SNR = 10 dB.

4.2.1. Some practical considerations

An important practical aspect with the update of the eigenvalues in MALASE algorithm is that it involves the inverse of the eigenvalues. In order to prevent possible numerical problems when the smallest estimated eigenvalue comes close to zero, eigenvalue updates in MALASE(K) are modified as in

$$\lambda_{k,n} = \lambda_{k,n-1} + \frac{\mu'}{(\lambda_{k,n-1} + \beta)^2} (|\xi_{k,n-1}|^2 - \lambda_{k,n-1}), \tag{26}$$

where $\beta > 0$.

An interesting possibility, for accelerating the convergence of the eigenspaces, without affecting the asymptotic performance of the algorithm, consists in setting a lower threshold, denoted $\theta_{\min}(> 0)$, for the rotation angle $\theta_n = \mu c_n$ that appears in the eigenvector update of MALASE(K). With this simple procedure, we avoid very small values of θ_n that can be obtained during the convergence phase of the eigenvalues and that would slow down the convergence of U_n. In addition, we are going to see in the next subsection that θ_n can be controlled efficiently to boost the initial convergence.

For very large values of the stepsize μ, the long term stability of the algorithm may not be satisfied. Then, a higher threshold $\theta_{\max}$ can be set to prevent the divergence of the algorithm. For instance, taking $\mu = \mu' = 0.02$ at SNR = 10 dB, the stability is ensured for $\theta_{\max} = 0.1$, without significantly modifying the initial convergence. But in fact, the range that we are going to consider for μ is much smaller than this and it will not be necessary to set any higher threshold for θ_n.

4.2.2. Initial convergence of err_n

First, we consider a simulation at low SNR (−5 dB). The curves are obtained from 10 averaged experiments, as well as all the following curves that will show the evolution of err_n. We see in Fig. 1 that the performance of MALASE is close to that of PROTEUS-2. However, it is possible to boost initial convergence very easily. Indeed, adding to θ_n a decreasing term such as $d_n = \exp(-n/100)$ enables much faster initial convergence of the algorithm without modifying its long term behavior. In Fig. 2, the high efficiency of this simple procedure clearly appears. Fig. 3 shows the evolution of err_n at SNR = 10 dB when using $\theta_n + d_n$ instead of θ_n. We obtain good performance for MALASE algorithm, despite the fact that the likelihood criterion is very ill-conditioned at such a high SNR.

4.2.3. Behavior of the eigenvalues

Figs. 4–6 show the evolution of the estimated eigenvalues for MALASE, PROTEUS-2 and LORAF-3 algorithms respectively, for SNR = 10 dB. The true eigenvalues are equal to 246, 222, 117, 59 and 1. Fast convergence speed is achieved with PROTEUS-2 algorithm, at the expense of a higher variance for the largest eigenvalues. For MALASE algorithm, faster convergence is achieved for the smallest eigenvalues, while the largest eigenvalues have not completely converged yet after 5000 iterations. At low SNR, similar convergence rate of all eigenvalues is achieved with MALASE, since they are not so spread.

We note that in accordance with the asymptotic convergence study the variance of all the

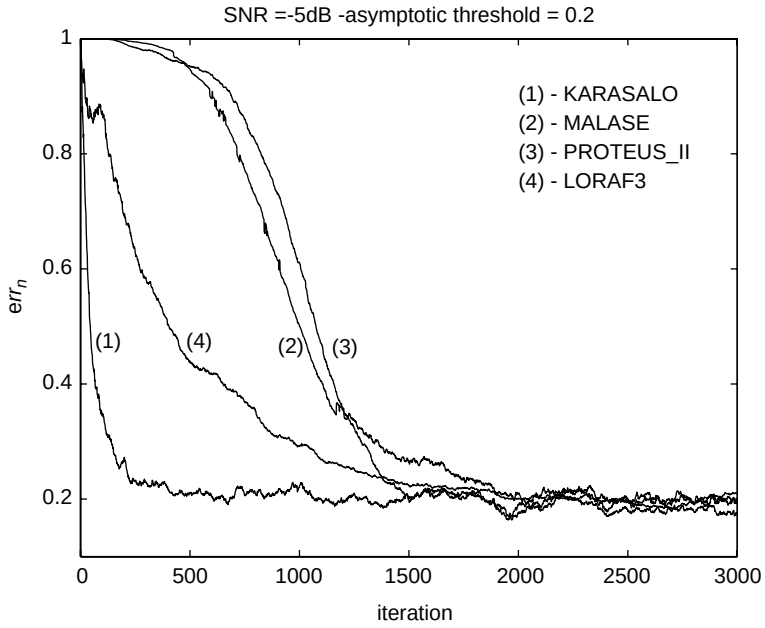

Fig. 1. Convergence of source subspace eigenvectors for SNR $= -5$ dB (see Table 1 for parameter values).

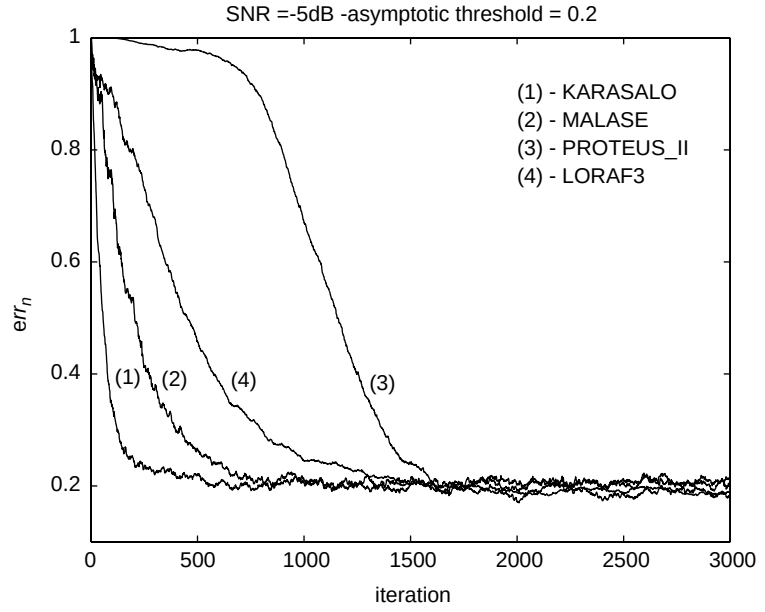

Fig. 2. Convergence of source subspace eigenvectors for SNR $= -5$ dB with accelerated initial convergence: $\theta_n \rightarrow \theta_n + d_n$ (Table 1).

eigenvalues is about the same with MALASE, except for the smallest one, which is smaller, due to the presence of the correcting term β in (26). On the contrary, for PROTEUS-2, the variance is much higher for large eigenvalues. This is easy to understand since for PROTEUS-2 the eigenvalues are updated from

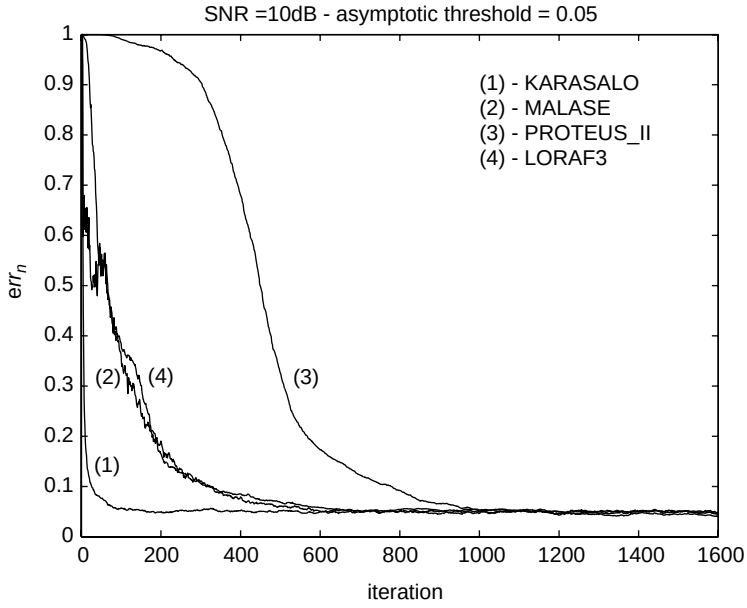

Fig. 3. Convergence of source subspace eigenvectors for SNR = 10 dB with accelerated initial convergence: $\theta_n \rightarrow \theta_n + d_n$ (Table 1).

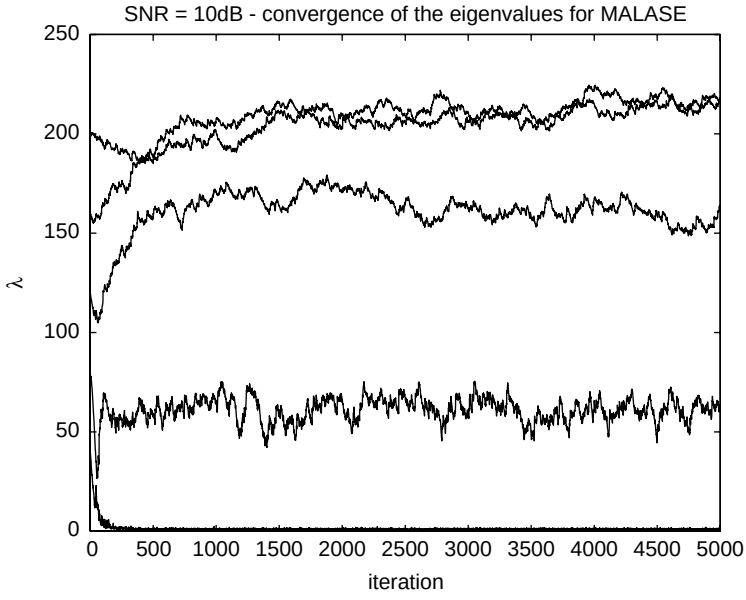

Fig. 4. Convergence of the eigenvalues for SNR = 10 dB: MALASE with accelerated initial convergence: $\theta_n \rightarrow \theta_n + d_n$ (Table 1).

(23), and for small μ the asymptotic variance for λ_k is $\sigma^2_{\lambda_k} = (\mu/2)\lambda_k^2$ for (23), while it is $\mu/2$ for the MALASE update.

With LORAF-3 the convergence results for the eigenvalues are not so good (Fig. 6): with this algorithm, the eigenvalues are obtained as the diagonal

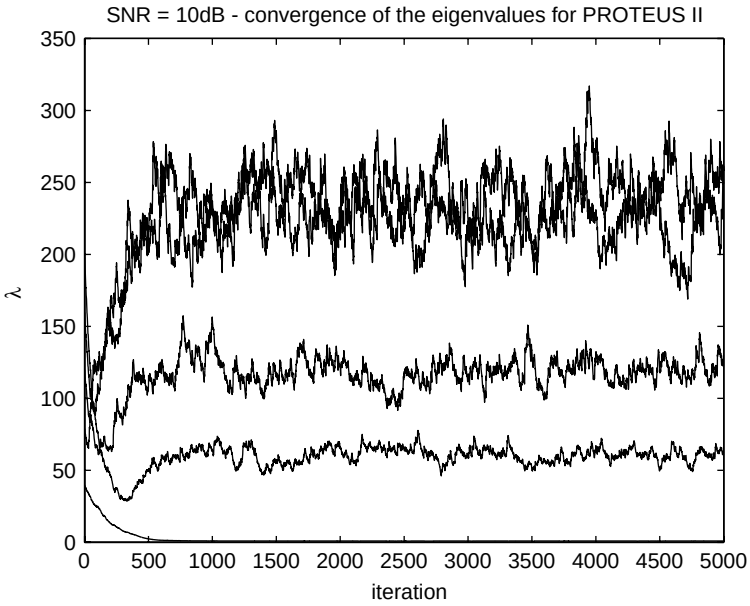

Fig. 5. Convergence of the eigenvalues for SNR = 10 dB: PROTEUS-2.

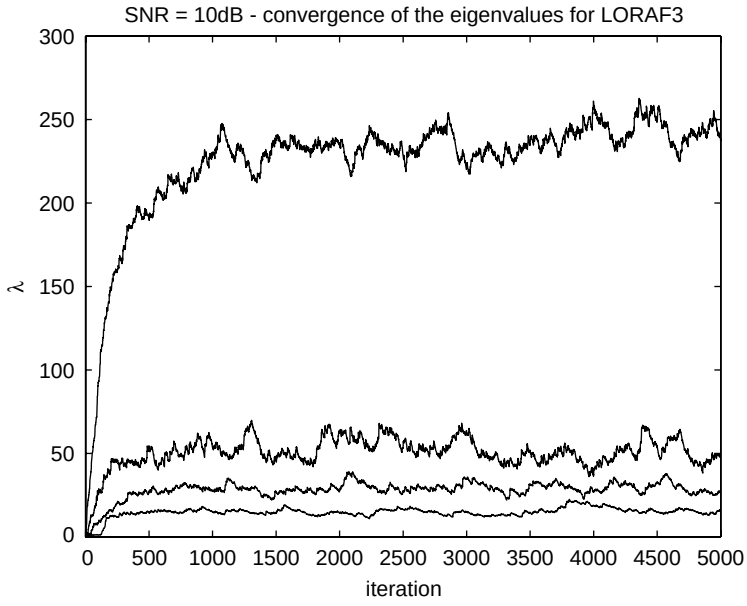

Fig. 6. Convergence of the eigenvalues for SNR = 10 dB: LORAF-3.

elements of the R matrix of a QR decomposition, as suggested in [26], but clearly only the largest eigenvalue is correctly estimated.

4.2.4. *Abrupt rotation of the eigenspace*

A popular way to check the robustness of a subspace tracking algorithm consists in applying a

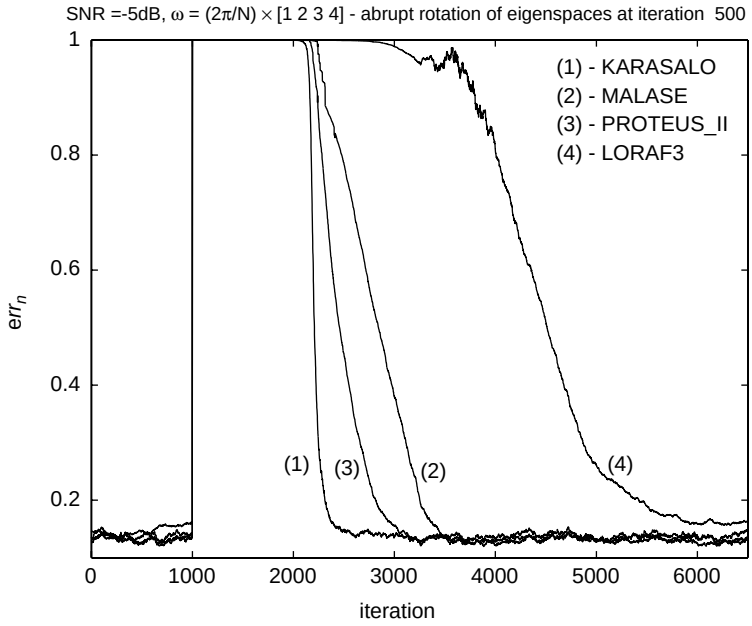

Fig. 7. Effect of an abrupt subspace rotation for SNR $= -5$ dB (Table 2).

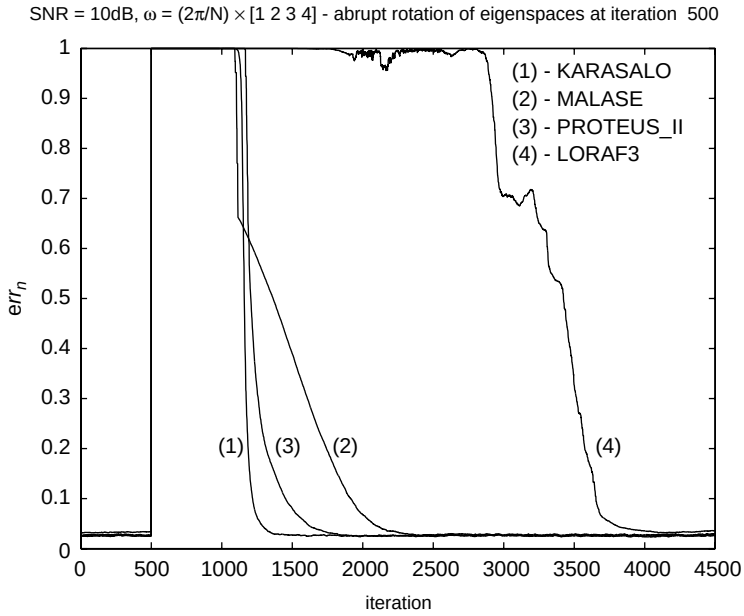

Fig. 8. Effect of an abrupt subspace rotation for SNR $= 10$ dB (Table 2).

sudden change of the true subspace after convergence, so that the new signal eigenspace is exactly orthornormal to the previous one [10]. We consider this kind of situation by changing the values $(\omega_k)_{k=1,p}$ by $(-\omega_k)_{k=1,p}$, where $\omega_k = 2k\pi/N$ $(k = 1, p)$. The results are given in Figs. 7 and 8 for SNR $= -5$ and 10 dB,

Table 2
Stepsizes for simulations where $\omega = (2\pi/16) \times [1, 2, 3, 4]$

	SNR	
	−5 dB	10 dB
Karasalo	$\mu = 0.0075$	$\mu = 0.01$
MALASE	$\mu = 0.0015$ $\mu' = 0.15$ $\theta_{\min} = 0.01$ $\beta = 1$	$\mu = 0.000065$ $\mu' = 300$ $\theta_{\min} = 0.003$ $\beta = 40$
PROTEUS-2	$\mu = 0.01$	$\mu = 0.01$
LORAF-3	$\alpha = 1 - 0.015$	$\alpha = 1 - 0.002$

respectively, where the eigenspace rotation is applied at iteration 500. The corresponding stepsizes are those in Table 2: they have been chosen again to enable similar asymptotic value for err_n. In both cases MALASE and PROTEUS-2 achieve much faster subspace recovery than LORAF-3.

4.2.5. *Abrupt deviation from orthonormality*

In order to check the influence of a lack of orthonormality of the estimated eigenvectors, a random perturbation is added to the matrix of the estimated eigenvectors after convergence. The ratio between the perturbation matrix norm and the initial matrix norm is 0.3. It appears (see Figs. 9 and 10) that the error is corrected efficiently by all the algorithms. In particular, this result shows that a lack of orthonormality is correctly suppressed by MALASE, despite the fact that it has been derived under an orthonormality assumption upon the eigenvectors. This suggests that MALASE algorithm initial convergence should be little sensitive to the lack of orthonormality of U_0. In fact, MALASE is robust to this phenomenon, especially when replacing θ_n by $\theta_n + d_n$: choosing random Gaussian variables with variances equal to one for the entries of U_0 yields the results in Figs. 11 and 12. In this situation, PROTEUS-2 has not yet converged after 2000 iterations and it has not been included in the figures. Comparing with Figs. 2 and 3 shows that LORAF-3 significantly suffers from such an initialization, while MALASE is not affected.

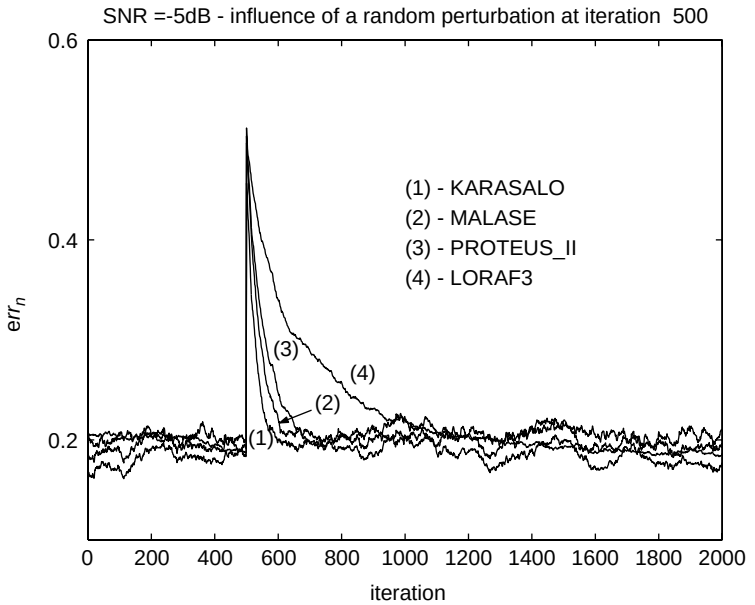

Fig. 9. Effect of a sudden loss of orthonormality of the eigenbasis for SNR = −5 dB (Table 1).

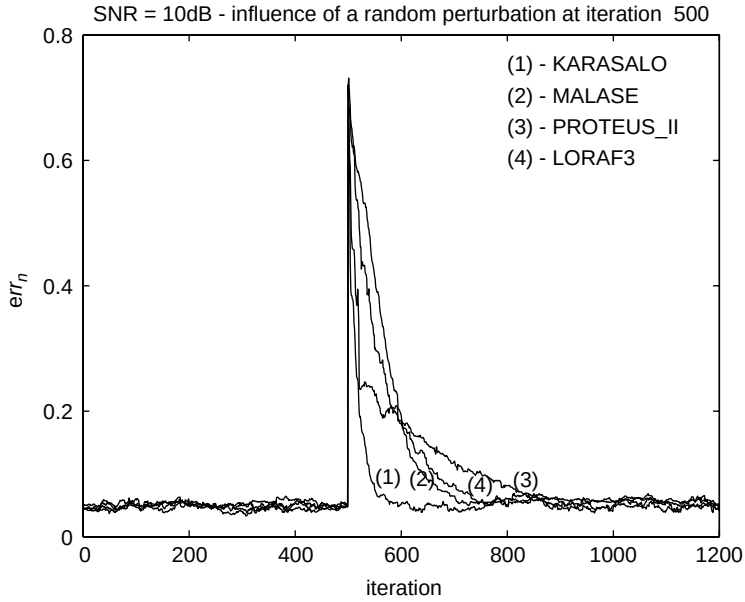

Fig. 10. Effect of a sudden loss of orthonormality of the eigenbasis for SNR = 10 dB (Table 1).

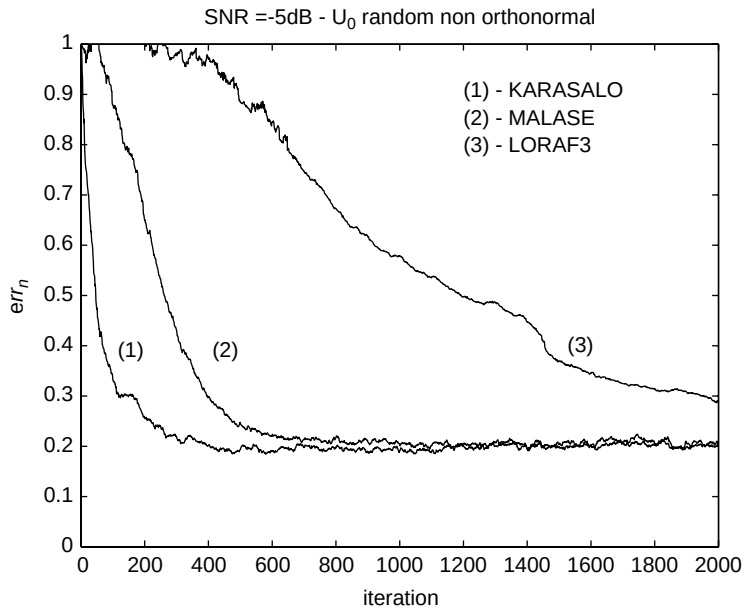

Fig. 11. Effect of a loss of orthonormality of U_0 for SNR = −5 dB (Table 1).

5. Conclusion

We presented a new technique to perform adaptive eigenvalue decomposition of a time varying covariance matrix, based on the statistically meaningful maximum likelihood criterion. We have shown that accounting for the orthonormality constraint upon the eigenvectors yields a simple iterative algorithm

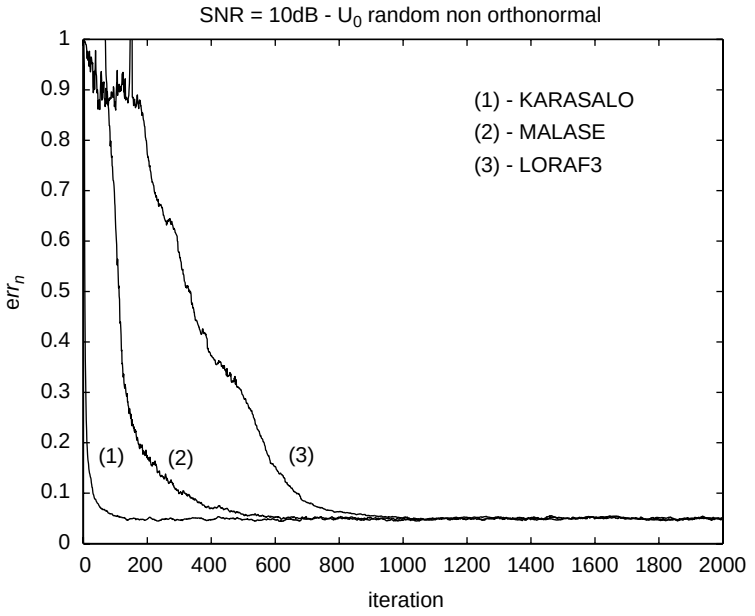

Fig. 12. Effect of a loss of orthonormality of U_0 for SNR = 10 dB (Table 1).

with nice features: desirable convergence properties are ensured, as well as orthonormality of the estimated eigenvectors, at the expense of small computational cost. We checked on examples the very good practical behavior of the MALASE algorithm, which can be understood in particular because it was derived without any first order approximation.

Acknowledgements

The authors are grateful to the anonymous reviewer who supplied them with Ref. [13] that presents time matrix algorithms.

Appendix A. Gradient based update of the ML criterion

For the likelihood criterion $\Phi(\mathbf{x}_n; U, \Lambda) = \sum_{k=1,N} \log \lambda_k + \mathbf{x}_n^{\mathrm{H}} U \Lambda^{-1} U^{\mathrm{H}} \mathbf{x}_n$, straightforward calculations show that the actualization of the EVD by means of a gradient algorithm is given by

$$\mathbf{y}_n = U_{n-1}^{\mathrm{H}} \mathbf{x}_n,$$

$$U_n = U_{n-1} - \mu \mathbf{x}_n \mathbf{y}_n^{\mathrm{H}} \Lambda_{n-1}^{-1},$$

$$\Lambda_n = \Lambda_{n-1} - \mu(\Lambda_{n-1}^{-1} - \Lambda_{n-1}^{-2} \operatorname{diag}(\operatorname{diag}(\mathbf{y}_n \mathbf{y}_n^{\mathrm{H}}))), \tag{A.1}$$

where μ is the stepsize of the algorithm. Unfortunately, this algorithm is not satisfactory. Indeed, let X be a wide sense stationary process, and (U, Λ) denote a stationary point of algorithm (A.1). Then, the update term of U_n must satisfy

$$E\{X_n X_n^{\mathrm{H}} U \Lambda^{-1}\} = R_X U \Lambda^{-1} = 0, \tag{A.2}$$

leading to $U\Lambda^{-1} = 0$. This proves that (U_n, Λ_n) does not converge to the parameters of the EVD of R_X.

Appendix B. Direct derivation of the algorithm

MALASE algorithm originates in the fact that the orthonormality of U_n is accounted for both in the computation of the gradient of the log-likelihood criterion and in the update of U_n. To see it more clearly, let us consider the differentiation of $\Phi(\mathbf{x}_n; U, \Lambda)$ with respect to U, with the unitary matrix constraint $UU^{\mathrm{H}} = I_N$ that yields $\mathrm{d}U^{\mathrm{H}} = -U^{\mathrm{H}} \mathrm{d}U U^{\mathrm{H}}$. Up to a second

order term, we have

$$\begin{aligned} &\mathrm{d}_U \Phi(\mathbf{x}_n; U, \Lambda) \\ &\quad = \Phi(\mathbf{x}_n; U + \mathrm{d}U, \Lambda) - \Phi(\mathbf{x}_n; U, \Lambda) \\ &\quad = \mathbf{x}_n^{\mathrm{H}}(\mathrm{d}U\Lambda^{-1}U^{\mathrm{H}} + U\Lambda^{-1}\,\mathrm{d}U^{\mathrm{H}})\mathbf{x}_n \\ &\quad = -\mathbf{y}_n^{\mathrm{H}}(\Lambda^{-1}U^{\mathrm{H}}\,\mathrm{d}U - U^{\mathrm{H}}\,\mathrm{d}U\Lambda^{-1})\mathbf{y}_n, \end{aligned} \tag{B.1}$$

with $\mathbf{y}_n = U^{\mathrm{H}}\mathbf{x}_n$. Then, it comes that under the unitary constraint over U, the matrix $\partial\Phi/\partial U$ with general term $[\partial\Phi(\mathbf{x}_n; U, \Lambda)/\partial U]_{i,j} = \partial\Phi/\partial[U]_{i,j}$ is given by

$$\frac{\partial \Phi}{\partial U} = -U(\Lambda^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda^{-1}). \tag{B.2}$$

Then the stochastic gradient update of U yields

$$\begin{aligned} U_n &= U_{n-1}(I + \mu(\Lambda^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda^{-1})) \\ &= U_{n-1}\exp(\mu(\Lambda^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda^{-1})) \\ &\quad + O(\mu^2). \end{aligned} \tag{B.3}$$

Forgetting the $O(\mu^2)$ term we get the proposed updating scheme that preserves orthonormality.

Let us remark that updating U_n in a way that involves the exponential of a skew-symmetric matrix is a property shared with some other stochastic algorithms intended to update a unitary matrix (see e.g. [6,8]). This is quite natural since the tangent plane to the unitary matrices manifold at point U is generated by matrices $\{UA; A = -A^{\mathrm{H}}\}$ and, as mentioned in Section 2.2, UA is the tangent vector at point U for the matrix function $U(t) = U\exp(tA)$ defined on the unitary matrices manifold [13].

Appendix C. Calculation of $\exp(\mu(\Lambda_{n-1}^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda_{n-1}^{-1}))$

Clearly,

$$[\mathbf{z}_n\ \mathbf{y}_n] = \begin{bmatrix} \dfrac{\mathbf{z}_n}{\|\mathbf{z}_n\|} & \dfrac{\mathbf{z}_n(\mathbf{z}_n^{\mathrm{H}}\mathbf{y}_n) - \mathbf{y}_n\|\mathbf{z}_n\|^2}{\|\mathbf{z}_n\|c_n} \end{bmatrix} \times \begin{pmatrix} \|\mathbf{z}_n\| & \dfrac{\mathbf{z}_n^{\mathrm{H}}\mathbf{y}_n}{\|\mathbf{z}_n\|} \\ 0 & \dfrac{-c_n}{\|\mathbf{z}_n\|} \end{pmatrix} = Q_n R_n \tag{C.1}$$

with $c_n = \sqrt{(\mathbf{z}_n^{\mathrm{H}}\mathbf{z}_n)(\mathbf{y}_n^{\mathrm{H}}\mathbf{y}_n) - (\mathbf{y}_n^{\mathrm{H}}\mathbf{z}_n)^2}$, and

$$\begin{aligned} &\exp(\mu(\Lambda_{n-1}^{-1}\mathbf{y}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{y}_n^{\mathrm{H}}\Lambda_{n-1}^{-1})) \\ &\quad = \exp(\mu(\mathbf{z}_n\mathbf{y}_n^{\mathrm{H}} - \mathbf{y}_n\mathbf{z}_n^{\mathrm{H}})) \\ &\quad = \exp(\mu[\mathbf{z}_n\ \mathbf{y}_n]\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}[\mathbf{z}_n\ \mathbf{y}_n]^{\mathrm{H}}) \\ &\quad = I + Q_n\left[\sum_{k=1,\infty}\frac{\mu^k}{k!}\left(R_n\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}R_n^{\mathrm{H}}\right)^k\right]Q_n^{\mathrm{H}} \\ &\quad = I + Q_n\begin{pmatrix} \cos\mu c_n - 1 & -\sin\mu c_n \\ \sin\mu c_n & \cos\mu c_n - 1 \end{pmatrix}Q_n^{\mathrm{H}}. \end{aligned} \tag{C.2}$$

References

[1] K. Abed-Meraim, A. Chkeif, Y. Hua, Fast orthogonal PAST algorithm, IEEE Signal Proc. Lett. 7 (March 2000) 60–62.

[2] H. Akaike, A new look at the statistical model identification, IEEE Trans. Automat. Control 19 (December 1974) 716–723.

[3] A. Barabell, Improving the resolution performance of eigenstructure-based direction-finding algorithms, Proceedings of the ICASSP'83, 1983, pp. 336–339.

[4] A. Benveniste, M. Metivier, P. Priouret, Adaptive Algorithms and Stochastic Approximation, Springer, Berlin, 1990.

[5] G. Bienvenu, L. Kopp, Optimality of high resolution array using the eigensystem approach, IEEE Trans. Acoust. Speech Signal Process. 31 (October 1983) 1235–1247.

[6] J.F. Cardoso, B.H. Laheld, Equivariant adaptive source separation, IEEE Trans. Signal Process. 44 (December 1996) 3017–3029.

[7] B. Champagne, Adaptive eigendecomposition of data covariance matrices based on first-order perturbations, IEEE Trans. Signal Process. 42 (October 1994) 2758–2770.

[8] B. Champagne, Q.G. Liu, A new family of EVD tracking algorithms using givens rotations, Proceedings of the ICASSP'96, 1996, pp. 2539–1542.

[9] B. Champagne, Q.G. Liu, Plane rotation-based EVD updating schemes for efficient subspace tracking, IEEE Trans. Signal Process. 46 (July 1998) 1886–1900.

[10] P. Comon, G.H. Golub, Tracking a few extreme singular values and vectors in signal process, Proc. IEEE 1978 (August 1990) 1327–1343.

[11] R.D. DeGroat, Noniterative subspace tracking, IEEE Trans. Signal Process. 40 (March 1992) 571–577.

[12] R.D. DeGroat, E.M. Dowling, D.A. Linebarger, Subspace tracking, in: V.K. Madisetti, D.B. Williams (Eds.), The Digital Signal Processing Handbook, CRC Press, Boca Raton, FL, 1998.

[13] J. Dehaene, Continuous time matrix algorithms, systolic algorithms and adaptive neural networks, Ph.D. Thesis, Katholieke Universiteit Leuven, October 1995.
[14] S. Erlich, R. Yao, Convergences of adaptative block simultaneous iteration method for eigenstructure decomposition, Signal Process. 37 (January 1994) 1–13.
[15] A. Gersho, R.M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers, Dordrecht, MA, 1995.
[16] G.H. Golub, C.F. Van Loan, Matrix Computation, 2nd Edition, Johns Hopkins Series in Mathematical Sciences, Johns Hopkins University Press, Baltimore, MD, 1990.
[17] I. Karasalo, Estimating the covariance matrix by signal subspace averaging, IEEE Trans. Signal Process. 37 (January 1986) 8–12.
[18] M. Moonen, Van Dooren, J. Vandewalle, A singular value decomposition algorithm for subspace tracking, SIAM Matrix Anal. Appl. 13 (October 1992) 1015–1038.
[19] E. Moulines, P. Duhamel, J.F. Cardoso, S. Mayrargue, Subspace methods for the blind identification of multichannel FIR filters, IEEE Trans. Signal Process. 43 (February 1995) 516–525.
[20] P.A. Pango, B. Champagne, On the efficient use of Givens rotation in SVD-based subspace tracking algorithms, Signal Process. 74 (1999) 253–277.
[21] P.A. Regalia, An adaptative unit norm filter with application to signal analysis and Karhunen–Loeve transformations, IEEE Trans. Circuits Systems 37 (May 1990) 646–649.
[22] C. Riou, T. Chonavel, Fast adaptive eigenvalue decomposition, a maximum likelihood approach, Proceedings of the ICASSP'97, 1997, pp. 3565–3568.
[23] C. Riou, T. Chonavel, P.Y. Cochet, Adaptive subspace estimation: application to moving sources localization and blind channel identification, Proceedings of the ICASSP'96, 1996, pp. 1649–1652.
[24] J. Rissanen, Modeling by the shortest data description, Automatica 14 (1978) 465–471.
[25] R.O. Schmidt, Multiple emitter location and signal parameter estimation, IEEE Trans. Antennas Propagation 34 (March 1986) 276–280.
[26] P. Strobach, Low rank adaptive filters, IEEE Trans. Signal Process. 44 (12) (December 1996) 2932–2947.
[27] H. Weyl, The Theory of Groups and Quantum Mechanics, Dover, New York, 1950.
[28] B. Yang, Subspace tracking based on the projection approach and the recursive least squares method, Proceedings of the ICASSP'93, 1993, pp. 145–148.
[29] B. Yang, Projection approximation subspace tracking, IEEE Trans. Signal Process. 43 (January 1995) 95–107.