Tracking a Few Extreme Singular Values and Vectors in Signal Processing

PIERRE COMON, MEMBER, IEEE AND GENE H. GOLUB, HONORARY MEMBER, IEEE

In various applications, it is necessary to keep track of a low-rank approximation of a covariance matrix, R(t), slowly varying with time. It is convenient to track the left singular vectors associated with the largest singular values of the triangular factor, L(t), of its Cholesky factorization. These algorithms are referred to as "square-root." The drawback of the EigenValue Decomposition (EVD) or the Singular Value Decomposition (SVD) is usually the volume of the computations. Various numerical methods carrying out this task are surveyed in this paper, and we show why this admittedly heavy computational burden is questionable in numerous situations and should be revised. Indeed, the complexity per eigenpair is generally a quadratic function of the problem size, but there exist faster algorithms whose complexity is linear. Finally, in order to make a choice among the large and fuzzy set of available techniques, comparisons are made based on computer simulations in a relevant signal processing context.

I. Introduction

Computing a few of the largest eigenvalues and associated eigenvectors is necessary to approximate a linear hermitian operator by another of lower rank. This occurs in the computation of principal components in data analysis [1], the approximate solution of huge systems in seismics, image processing and coding [2], or model reduction [3], Karhunen-Loeve truncated expansions in estimation and detection theory [4] and in pattern recognition [5], minimal realizations in system theory [6], [7], dominant eigenmodes enhancement in mechanics, resolution of shift-invariant differential equations [8], geophysics [9], [10], spectral analysis [11]-[13], or antenna array processing [14], [15]. In other applications the Generalized Eigenvalue Problem (GEP) is frequently solved; the most well known case occurs in mechanics [16]. But it also arises in signal subspace methods when the background noise is not homogeneous and isotropic [14], [17], [18]. A recent formulation of this probem using a Total Least Squares (TLS) approach (see [25] for definition) requires the solution of a large generalized Singular Value Decomposition (SVD) problem [19], [20]. For the sake of convenience, the generalized problems will not be discussed in this paper.

Manuscript received February 8, 1989; revised May 7, 1990. This work was supported in part by the U.S. Army under Contract DAAL 03-87-k0095.

P. Comon is with THOMSON-SINTRA, BP 53, F-06801, Cagnes Sur Mer Cedex, France.

G. H. Golub is with the Department of Computer Science, Stanford University, Stanford, CA 94305, USA.

IEEE Log Number 9037865.

The matrices involved in these problems are hermitian (or real symmetric) positive definite, and frequently structured. For instance they can be sparse, banded [21], [22], Toeplitz or Hankel [23], [24], possibly by blocks. It is desirable to take advantage of the existing structure of the matrices when computing the eigenpairs. With this purpose, it will be shown subsequently why the so-called Lanczos algorithms [22], [25] are useful, and may be considered as reference methods for adaptive computation. Application to the so-called signal subspace methods, and particularly to MUSIC [14], [15] and ESPRIT [19], [20] methods, will be focused in this paper. As mentioned above, the smallest eigenvalue may be also of interest, or equivalently the smallest singular value when operating on the data matrix directly, and this computation may be handled as well.

Moreover, the problem of tracking these eigenpairs, when the matrix is slowly varying in time, turns out to be central to these types of adaptive antenna array processing techniques, and some numerical methods have already been proposed [26]-[29]. Some of these methods take into account the fact that the hermitian matrix varies by a rank one modification, from one time step to the next, in order to speed up the computational time. Therefore, other well-known numerical methods could be used for this purpose [25], [30], [31], but they turn out to be less attractive under their original form since they require computing all the eigenpairs, and thus increase the computational complexity. The adaptive methods described in this paper are grouped into two families, according to their complexity. In the first one, the algorithms require $O(dn^2)$ operations to compute the d extreme eigenpairs of a full matrix and $O(dn \log n)$ for Toeplitz or Hankel matrices, whereas they require $O(nd^2)$ flops in the second. Algorithms of the second family converge much more slowly, but it turns out to be sufficient when the eigenpairs of interest vary slowly with time (e.g., see algorithms F2 and G). In the first family, the method giving the best accuracy/complexity ratio seems to be the Lanczos-based approach.

A table of contents is included below to enable the reader to quickly locate various techniques:

0018-9219/90/0800-1327$01.00 © 1990 IEEE

III. **Basic algorithms**
   - A. Standard Iterative Algorithms
   - B. Appending a Column to a Matrix (Bunch *et al.*)
   - C. Lanczos-type Bidiagonalization Algorithms (block-Lanczos, BSR).
   - D. Ritz Triplets and Accuracy in Exact Arithmetic
   - E. Block-Bidiagonalization
   - F. Reorthogonalization Versus Size Reduction

IV. **Adaptive algorithms with $O(m^2d)$ complexity**
   - A. Lanczos-based Recursions (algorithm A)
   - B. Subspace Iteration with Ritz Acceleration (algorithm SIR)
   - C. Gradient-based Algorithms (algorithm C)
   - D. A Hybrid Method (algorithms D1 and D2)

V. **Adaptive algorithms with $O(md^2)$ complexities**
   - E. Approximate Subspace Iteration (algorithm E)
   - F. Modeling of Class F (algorithms F1 and F2)
   - G. Modeling of Class G (algorithm G, Karasalo, Schreiber)
   - H. Instantaneous Stochastic Gradient Methods (algorithms H1 and H2)

VI. **Computer simulations**

Appendix: Complexities of QR and SVD Factorizations

In this paper, we frequently refer to matrix decompositions, which are described in [25]. The programs for performing them are available in *Linpack* and *Eispack* [32], [33]. We have found it useful and concise to express our algorithms in a *Matlab*-like language. See [34] for further details on this helpful package.

II. The Observation Model in Antenna Array Processing

A. Observation Model

The observation model generally assumed in antenna array processing has the following form:

$$r(k) = A(k)\, s(k) + n(k). \tag{1}$$

In this linear statistical model, the $n \times 1$ vector $r(k)$ denotes the observation, $A(k)$ is a deterministic $n \times d$ matrix, $s(k)$ is a random $d \times 1$ vector (the "source" vector) and $n(k)$ stands for a noise disturbance; k denotes a discrete frequency variable [15], [14]. In these problems, the relevant unknowns to compute are as follows: (1) the matrices $A(k)$ under the constraint that they belong to a given parametrized manifold; (2) the covariance matrix of the signal component (the entries in $s(k)$), sometimes assumed to be diagonal; (3) the covariance matrix of the noise $n(k)$ (4) possibly an estimate of the signal vector $s(k)$ called the "signal copy" is also of interest; (5) sometimes, the size d of the vector $s(k)$ may be unknown, and even varying with time.

In order to solve this estimation problem, an additional knowledge of the noise covariance matrix $N(k) = E\{n(k)\, n(k)^H\}$ is required ($(^H)$ denoting hermitian transposition). Generally, $N(t)$ is assumed to be of the form:

$$N(k) = p^2(k)\, G(k) \tag{2}$$

where $p^2(k)$ is the noise level (a scalar unknown parameter), and $G(k)$ is a known fixed positive definite matrix which has been estimated beforehand with the help of a parametric model [35]. Since signals and noise are uncorrelated, the following relation between covariance matrices may be deduced:

$$\bar{R}(k) = E\{r(k)\, r(k)^H\} = A(k)\, \bar{S}(k)\, A^H(k) + p^2(k)\, G(k). \tag{3}$$

A spatial prewhitening of the data allows us to deal only with the case $G(k) = I$, the identity matrix [14], [15].

We see that estimating the covariance matrix of the signals consists of approximating the matrix $\bar{R}(k)$ by another of lower rank, namely $A(k)\, S(k)\, A^H(k)$, provided we eliminate the constraints upon the matrices $A(k)$. In the so-called subspace methods (esp. MUSIC), the determination of the Directions Of Arrival (DOA) proceeds in two steps [14], [15]. In order to remove the noise effect and estimate the signal subspace spanned by the columns of A, a low rank approximate to $\bar{R}(k)$ is found, denoted $\bar{R}^s(k)$. Second, the parametrized antenna manifold, A, is searched in order to pick up the few admissible values for the directions in the signal subspace (linear space spanned by the columns of A), and then deduce the parameters of interest (including the DOAs). We do not go further into these details since it is out of the scope of the paper, and restrict ourselves to the computation of $\bar{R}^s(k)$. The point to consider now is the way the matrix varies.

B. Covariance Matrix Updates

For notational convenience, we drop the variable k, it being understood that every variable depends on the frequency, or equivalently, that the processing acts on narrowband objects. Special techniques matched to wide-band DOA estimation [36] will not be considered in this paper, although they have many similarities. Now, in practice the matrix R is itself unknown so that an estimate of it must be used. Commonly, estimates of the covariance matrix are made as an average of outer products of observations such:

$$R = c_M \sum_{\mu=1}^{M} r_\mu r_\mu^H, \qquad c_M = \text{a given scalar},$$

where the r_μ are different snapshots of observations satisfying the model (1). Now, the goal is to track a time-varying system, so that the covariance function also depends on a time matrix. However, we shall still use the "covariance matrix" $\bar{R}$, noting that it approximates the second order moment of the observed process on a restricted time range. Additionally, we have not at our disposal several snapshots of the same phenomenon, but a single realization of the random vectors, $r(t)$, having statistics close to each other. For instance, among the possible estimates of the matrix $\bar{R}$, that appears convenient to write a time-dependent form $\bar{R}(t)$. One is widely used and has the form:

$$R(t) = \beta(t)\, R(t-1) + \alpha(t)\, r(t)\, r(t)^H. \tag{4}$$

In other words, at each time step t, the matrix $R(t)$ is updated via an additive rank-one modification. This averaging is referred to as exponential (since it indeed is when $\alpha(t)$ is constant); $R(t)$ contains an infinite memory of the past of the system, which is weighted by the coefficients $\alpha(t)$ and $\beta(t-p)$, $p \in \mathbf{N}$. The better these coefficients are chosen, the closer the covariance $R(t)$ to the local "true" covariance $\bar{R}(t)$. Our purpose is not to propose better estimates (and there are indeed other ways of defining the matrix R), but describe how to track extreme eigenpairs of a matrix $R(t)$ which is subject to small variations with time.

C. Use of the Data Matrix

Consider the Cholesky decomposition $R(t) = L(t)\, L(t)^H$ where $L(t)$ is lower triangular; then, the eigenpairs of the hermitian matrix $R(t)$ correspond to the left singular pairs of the triangular matrix $L(t)$. It is convenient to give an updat-

ing formula for matrix $L(t)$, that is equivalent to (4). Let H be the upper $(n + 1) \times n$ Hessenberg matrix defined by:

$$H(t)^H = [\sqrt{\alpha}(t)\, r(t),\ \sqrt{\beta}(t)\, L(t-1)]. \tag{5}$$

It is easily seen that $R(t) = H(t)^H H(t)$. We desire $L(t)$ to be lower triangular and to have the same dimension as $R(t)$. We generate $L(t)$ such that $H(t)^H Q(t) = [L(t), 0]$, where $Q(t)$ is a unitary transformation which eliminates the superdiagonal of $H(t)^H$. A QR factorization of the Hessenberg matrix $H(t)$ is calculated, and this takes order of $2n^2$ multiplications and n^2 additions using Jacobi rotations since the matrix $Q(t)$ is not required [25; p. 220].

It is in fact more numerically stable to update $L(t)$ rather than $R(t)$ [37]. The idea of using a square root of the matrix $R(t)$ is not new to the signal processing audience. In fact, this is used in state-space algorithms [38]–[40] to preserve the positivity of the innovation matrix, and improve the conditioning and rounding errors. In addition, it has been recently used in fast transversal filters [41], [42] to reduce the complexity.

D. Operation Counts

The number of floating point operations (flops) is defined as the number of multiplications, and for the sake of simplicity, we will make no distinctions between real and complex numbers. For instance, the product between a complex $n \times 1$ vector and a scalar (either real or complex) will be said to require n (complex) flops. In other words, the operation count is carried out as if all the numbers involved were real. In order to have an idea of the volume of computation in the algorithms presented in terms of real flops, one may consider that one complex flop is worth three or four real flops. The result will be slightly pessimistic.

III. Basic Algorithms

In this section, we shall focus on the computation of singular triplets of a fixed matrix (viz, singular values and associated left and right singular vectors). The basic algorithms are briefly reviewed, and the problems in using them for computing a few extreme triplets are emphasized.

A. Standard Iterative Algorithms

The Golub-Kahan-Reinsch algorithm has become a standard method for computing the SVD of a matrix on a serial computer [43]. The Kogbetliantz and Jacobi algorithms used earlier are also of great interest again after a long period because they can be often parallelized. Nevertheless, these three iterative algorithms are not well suited for the computation of a restricted subset of singular values. Thus, other procedures must be sought. Since $H(t)$ has been perturbed as indicated by (5), and in particular if $\alpha(t)$ is close to zero, one can attempt to compute the new set of singular triplets by noting that they have been only slightly perturbed. But this is not necessarily satisfied in practice, especially if nonstationary phenomena are tracked (one could have $\alpha(t) = 0.2$). Moreover, in order to compute the new singular triplets of the perturbed matrix, the knowledge of all the singular triplets of the original matrix is required [44; chap. 2], [45]. If we have the goal of tracking, the complete eigenstructure must be computed at each time step. These two remarks do a disservice to this approach. A second approach to speed up the determination of some extreme singular values is to take into account the rank-one modification, and this is developed in the next subsection.

B. Appending a Column to a Matrix

The rank-one updating of the eigenproblem and related special algorithms have been proposed and studied in the symmetric case [30], [31]. The principle is summarized as follows. Let:

$$M = B\Lambda B^H, \text{ and } R = M + \rho^2 uu^H,$$

where u is of unit norm and B is unitary.

We seek the eigenvectors x and eigenvalues μ such that:

$$(B\Lambda B^H + \rho^2 uu^H)x = \mu x.$$

Equivalently, denoting $y = B^H x$ and $z = B^H u$, we wish to compute the pairs $\{y, \mu\}$ such that:

$$(\Lambda + \rho^2 zz^H)y = \mu y.$$

This leads to the system:

$$(\Lambda - \mu I + \rho^2 zz^H)y = 0. \tag{6}$$

The eigenvalues μ_i may be obtained from the secular equation by finding the roots of the function [30]:

$$f(\mu) = 1 + \rho^2 \sum_{j=1}^{n} \frac{z_j^2}{(\lambda_j - \mu)} \tag{7}$$

where the z_j's denote the entries of z. The computation of one eigenvalue takes roughly $O(n)$ multiplications, and various algorithms have been proposed [30], [31]. Once the eigenvalues are known, the eigenvectors can be computed via inverse iteration [45b]. This awkward procedure would take $O(n^3)$ multiplications which is too much. But there is a better way to do this computation. The relation (6) may be rewritten as:

$$(\Lambda - \mu_i I)y_i = \gamma_i z; \text{ where } \gamma_i = -\rho^2 z^H y_i,\ 1 \le i \le d.$$

This yields immediately:

$$y_i = \gamma_i (\Lambda - \mu_i I)^{-1} z. \tag{8a}$$

Ignoring the scalar, γ_i, the eigenvector is normalized with $\|y_i\|$,

$$\|y_i\|^2 = z^H (\Lambda - \mu_i I)^{-2} z = f'(\mu_i)/\rho^2, \tag{8b}$$

which (except for the square root) is previously computed in the root finder part of the abovementioned method. The complexity of the computation of d eigenvectors, y_i, from z and μ_i takes about $2dn$ multiplications. The determination of the eigenvectors x_i of the original matrix requires one additional matrix-vector product. Therefore, computing d eigenpairs of a matrix perturbed by a rank-one modification requires $O(dn^2)$ multiplications. Note that if the smallest eigenvalue were repeated $(n - d)$ times, deflation would be possible and would decrease the complexity to $O(d^2 n)$. This can be exploited in adaptive algorithms of class G [46].

As explained in the previous section, adding a rank-one modification to an hermitian matrix has the same effect as appending a column to the triangular matrix of its Cholesky factorization. Thus, a similar algorithm can be derived for the update of the left singular pairs, the determination of the right singular vector being more complicated. Let N be

a general square matrix, and consider the SVD of N:

$$N = B\Delta C^H, \text{ with } \Delta = \text{diag}\{\delta_1, \cdots, \delta_n\}, B, C \text{ unitary},$$

$$H^H = [\rho u, N], \text{ and } [L, 0] = H^H Q,$$

where Q denotes a unitary matrix.

According to results in Section II, L is thus the updated square matrix. The singular values of L, σ_i, are again the solutions of the secular equation

$$g(\sigma^2) = 1 + \rho^2 \sum_{j=1}^{n} \frac{z_j^2}{(\delta_j^2 - \sigma^2)} = 0$$

and the left singular vectors of L may be obtained by solving the diagonal system

$$(\Delta\Delta^H - \sigma_i^2 I) y_i = z \tag{9}$$

and then computing $x = By/\|y\|$. The first step has $O(n)$ complexity and the second $O(n^2)$.

The complexity of the algorithms proposed above is attractive, but misleading. In fact, computing d left singular pairs takes $O(dn^2)$ operations, but requires the knowledge of all the left singular pairs before perturbation (because of the use of B) [31], [47]. In other words, these procedures cannot apply to successive rank-one modifications without saving all the previous updates. The same remark holds for updates of different nature [48], [49]. So, let us turn to other methods. Another efficient way to compute some singular triplets of a matrix, given ad hoc initial guesses, is to minimize $\|LL^H x - \mu x\|^2$ by using an iterative method. A good way to do this is to use the conjugate gradient algorithm, which leads to the Lanczos procedures.

C. The Lanczos-Type Bidiagonalization Algorithm

The Lanczos algorithms for symmetric matrices were introduced in the 1950s and are described, for instance, in [22], [25], [50]. The algorithms have received more attention following Paige's results [51], and have been studied in details during the last decade [52]. Thus there is no need to introduce the methods in detail here. The main advantages of the Lanczos methods come from the following: (1) the original matrix is not overwritten and (2) little storage is required since only matrix-vector products are computed. This makes the Lanczos procedures interesting for large matrices, especially if they are sparse or if there exist fast routines for computing matrix-vector products, such as for Toeplitz or Hankel matrices. The running of a Lanczos procedure applied to a symmetric matrix, R, yields a tridiagonal matrix, T; extreme eigenvalues close to those of the original matrix can be calculated from a submatrix of T, T°. Approximations to the eigevectors of R require the storage of the Lanczos vectors generated, and the computation of the eigenvectors of the matrix T° (which can be obtained cheaply by inverse iteration). These approximants are called Ritz vectors. The other advantage is that in exact arithmetic, the eigenpairs of T coincide with those of M if at most n iterations are performed. In other words, the algorithm terminates as soon as n (or fewer) orthogonal Lanczos vectors have been generated. In each step, the complexity is dominated by one matrix-vector product. If K steps are run, this yields a global complexity of $O(Kn \log n)$ for Toeplitz or Hankel matrices. Actually, because of roundoff errors, a loss of orthogonality among the computed Lanczos vectors occurs when some eigenpairs have already converged, and several undesirable phenomena may be observed [22], [25], [52]. This is usually coped with by introducing a reorthogonalization procedure which, of course, increases the complexity. If this is not done, then spurious eigenvalues may appear, and it is necessary to detect them. We will discuss this issue later.

In the applications to signal subspace methods, we seek the largest eigenvalues, and they are often well separated from the others. In this case, they will emerge before n iterations are performed, and typically after $O(1)$ iterations [53], [54]. We wish to make use of this in the design of our procedure. As pointed out earlier, we prefer to work on the data matrix directly and compute its singular triplets rather than the eigenpairs of the covariance matrix. In this case, a bidiagonalization may be carried out instead of a tridiagonalization. The algorithm, referred to as the Golub-Kahan algorithm is summarized in the following steps [55]:

Bidiagonalization of a $m \times n$ matrix L: (10)
Start with a given vector of size m, $u(1)$
% initialization
Normalize $u(1)$;
$tr = L^H * u(1)$; $a(1) = \text{norm}(tr)$;
$v(1) = tr/a(1)$;
% next steps **Complexity**
for $k = 1: K - 1$
  $tl = L^* v(k) - a(k)^* u(k)$; $b(k) = \text{norm}(tl)$; ***mn + 2m***
  $u(k+1) = tl/b(k)$; ***m***
  $tr = L^H * u(k+1) - b(k)^* v(k)$; $a(k+1) = \text{norm}(tr)$; ***mn + 2n***
  $v(k+1) = tr/a(k+1)$; ***n***
end

In the above algorithm, notations tr and tl stand for *temporary right* and *temporary left* vectors. The $k \times k$ bidiagonal matrix generated is then defined by:

$$B(k) = \begin{pmatrix} a(1) & & & \mathbf{0} \\ b(1) & . & & \\ & . & . & \\ \mathbf{0} & & b(k-1) & a(k) \end{pmatrix}$$

The algorithm above is more convenient if $m \leq n$. But, if $m \geq n$, a similar algorithm may be used, starting with an initial right vector $v(1)$, and yielding an upper bidiagonal matrix. One can deal with L or L^H; in our case, either one or the other starting vector is acceptable since we deal with a square matrix. The Lanczos bidiagonalization was first introduced by Golub and Kahan [55]; some useful properties were later pointed out by Paige [56].

D. Ritz Triplets and Accuracy in Exact Arithmetic

For the sake of convenience, define the matrix

$$\tilde{B}(k) = \begin{pmatrix} B(k) \\ 0 : b(k) \end{pmatrix}; \text{ note that } B(k) = \begin{pmatrix} & & : & 0 \\ & \tilde{B}(k-1) & : & \cdots \\ & & : & a(k) \end{pmatrix}.$$

In the recursion, $a(k)$ and $b(k)$ are in turn computed so that $B(k)$ and $\tilde{B}(k)$ are consequently determined. Let $\tilde{\Pi}(k)\ \tilde{\Theta}(k)\ \tilde{\Psi}(k)^H$ and $\Pi(k)\ \Theta(k)\ \Psi(k)^H$ be the SVD of $\tilde{B}(k)$ and $B(k)$,

respectively; $\tilde{\Theta}(k)$ being a $k \times (k-1)$ matrix with one row of zeros. At any step k, the following relations are always satisfied, and characterize the recursion of the Lanczos bidiagonalization algorithm:

$$L^H U(k) = V(k)\, B(k)^H \quad \text{and} \quad LV(k) = U(k+1)\, \tilde{B}(k) \tag{11}$$

where $U(k)$ and $V(k)$ represent the matrices of stacked left and right Lanczos vectors, $u(k)$ and $v(k)$. Substituting the SVD of $B(k)$ in (11) gives:

$$\Pi(k)^H\, U(k)^H\, LV(k)\, \Psi(k) = \Theta(k),$$

which shows that the k vectors contained in the matrices

$$W(k) = U(k)\, \Pi(k) \quad \text{and} \quad X(k) = V(k)\, \Psi(k) \tag{12}$$

are approximations to left and right singular vectors of L. The k columns in $W(k)$ and $X(k)$ are called left and right Ritz vectors, respectively, whereas the diagonal entries in $\Theta(k)$ are the Ritz values. Let θ_i, w_i and x_i denote the ith Ritz value, left and right Ritz vectors respectively computed at step $k-1$. Then the left and right Ritz pairs obtained at step $k-1$ in the algorithm given by (10) satisfy in exact arithmetic the accuracy equation [18]:

$$\|LL^H w_i - \theta_i^2 w_i\| = \sqrt{a(k)^2 + b(k)^2}\,|a(k)|\,|\tilde{\Pi}(k-1)_{ki}|$$
$$\forall i \le k-1$$
$$\|L^H L x_i - \theta_i^2 x_i\| = \sqrt{a(k)^2 + b(k)^2}\,|b(k)|\,|\Psi(k)_{ki}|$$
$$\forall i \le k \tag{13}$$

This extends the well known result of the symmetric case [25; p. 327]. Equations (13) show that if $a(k)$ or $b(k)$ are negligibly small, then the corresponding Ritz singular pairs have converged. Note that a convergence may also occur if some entry in the kth row of $\tilde{\Pi}(k-1)$ or $\Psi(k)$ is small.

In the Lanczos algorithms, extreme eigenpairs of LL^H will emerge first if they are isolated. However in signal processing the smallest eigenvalues often represent the noise covariance, and are quite clustered. For this reason, only largest eigenpairs will emerge in general. A possible means to compute the smallest eigenvalues is to consider formally the matrix L^{-1}: the smallest left Ritz pairs may be computed by solving two linear systems of the form $L^H * tr = u$ and $L * tl = v$ at each step k, instead of performing two matrix-vector products as described in the bidiagonalization algorithm (10). This requires a slightly larger computational time, still of order $O(n^2)$ multiplications, but the success depends on the eigenvalue pattern and is therefore not guaranteed.

E. Block-Bidiagonalization

Now, if we wish to compute several singular triplets, and if we have good initial guesses for more than one set, we can use a block recursion. Unlike the single-vector recursion, the block recursion can deal with multiple singular values. The block recursions are defined in a very similar manner by the algorithm given below. Let b_s denote the block size (number of starting vectors).

Block-bidiagonalization of a $m \times n$ matrix L: (14)
Start with a given block of size $m \times b_s$, $u(1)$.
% initialization
Orthonormalize $u(1)$;
$tr = L^H * u(1)$;
$[v(1), a(1)] = qr(tr)$; QR factorization.

% next steps	**Complexity**
for $k = 1: K-1$	
$tl = L * v(k) - u(k) * a(k)^H$;	$\boldsymbol{mnb_s + mb_s(b_s+1)/2}$
% QR factorization:	
$[u(k+1), b(k)] = qr(tl)$;	$\boldsymbol{2mb_s^2 - 2b_s^3/3}$
$tr = L^H * u(k+1) - v(k) * b(k)^H$;	$\boldsymbol{mnb_s + nb_s(b_s+1)/2}$
% QR factorization:	
$[v(k+1), a(k+1)] = qr(tr)$;	$\boldsymbol{2nb_s^2 - 2b_s^3/3}$
end	

In the algorithm above, the QR factorization of a rectangular $m \times b_s$ matrix provides a set of orthonormal vectors, Q, $Q^HQ = I$, and a $b_s \times b_s$ upper triangular matrix, R. This is the "economic version" of the QR factorization, whose complexity is $O(2mb_s^2 - 2b_s^3/3)$, instead of $O(2m^2b_s - mb_s^2 + b_s^3/3)$ as in the standard QR factorization [25; p. 148] (see Table 2). One of the interesting aspects in using a QR factorization is that the $(K * b_s) \times (K * b_s)$ block-bidiagonal matrix generated is banded and of bandwidth b_s [57]:

$$B(K) = \begin{pmatrix} a(1)^H & & & \mathbf{0} \\ b(1) & \cdot & & \\ & \cdot & \cdot & \\ \mathbf{0} & & b(K-1) & a(K)^H \end{pmatrix} \tag{15}$$

so that its singular values are obtainable in $O(K)$ multiplications.

F. Reorthogonalization Versus Size Reduction

In the single-vector recursions described above, it is clear that whenever some $a(k)$ or $b(k)$ in the bidiagonal matrix are very small, the result is ill-determined and may give an inaccurate Lanczos vector. Then there is a loss of orthogonality between the vector generated and the previous Lanczos vectors. These cancellations accumulate because the Lanczos recursion is not well determined. As a result, some spurious singular values may appear, as the number of iterations increases. Obviously, this problem also arises in the block-recursion whenever the triangular blocks $a(k)$ or $b(k)$ are not of full rank. Several ways have been proposed in the literature to cope with this problem. For instance in [22], various strategies are proposed in the symmetric case to identify the spurious eigenvalues. In the case of the computation of a few singular triplets, it turns out that a partial reorthogonalization could do very well, but would be computationally awkward. In fact, as equations (13) show, if the terms $a(k)$ or $b(k)$ are small, then there is a Ritz triplet that has converged. Similarly, in the block bidiagonalization, matrices $a(k)$ or $b(k)$ may be singular only if one of the Ritz triplet has converged. Instead of trying to reorthogonalize the set of b_s Lanczos vectors obtained, it is more convenient to stop the recursion on the Ritz triplet that has converged. We call this procedure "size reduction." When the size reaches zero, all the b_s Ritz triplets have converged, and the algorithm terminates. Recalling the notations used in Section III-D, $U(k)$ (resp. V) denotes the matrix made up of Lanczos vectors $\{u(1), \cdots u(k)\}$ (resp. v).

Suppose that $tl = qu * b$ is a QR factorization of tl, where qu is $m \times b_s$ and b is of rank $(b_s - 1)$. Since the triangular matrix b is singular, one of its diagonal entries must be zero. It is thus possible to easily detect rank-deficiency. The fact that tl is not of full rank indicates that a Ritz triplet has con-

verged. Then one should stop the recursion on the corresponding Lanczos vector (but only this one). As soon as the size of the block reduces to zero, the algorithm terminates.

The way to do this is to use a "Reduced Rank QR" (RRQR) factorization. The RRQR factorization of a rectangular $m \times b_s$ matrix of rank r is defined by the product of a semi-unitary matrix of size $m \times r$ and a $r \times b_s$ full rank upper triangular matrix. This factorization is always possible, and may be obtained via a Householder QR factorization with pivoting. The size reduction procedure may be run by using a $RRQR$ factorization instead of a QR as in algorithm (14). The understanding of this procedure is easier when looking at the complete algorithm (17):

Block-bidiagonalization of a $m \times n$ matrix L,
with Size Reduction (BSR). (17)
Start with a given block of size $m \times b_s$, $u(1)$.
% initialization
Orthonormalize $u(1)$;
$tr = L^H * u(1)$;
$[qv, a(1)] = rrqr(tr)$; % RRQR factorization.
$[r, b_s] = \text{size}(a(1))$;
$v(1) = qv$;
% next steps
for $k = 1: K - 1$
  $tl = L * v(k) - u(k) * a(k)^H$;
  $[qu, b(k)] = rrqr(tl)$; % RRQR factorization.
  $[r, b_s] = \text{size}(b(k))$;
  $u(k + 1) = qu$;
  if $r < 1$, stop
  $tr = L^H * u(k + 1) - v(k) * b(k)^H$;
  $[qv, a(k + 1)] = rrqr(tr)$; % RRQR factorization.
  $[r, b_s] = \text{size}(a(k + 1))$;
  $v(k + 1) = qv$;
  if $r < 1$, stop
end

For instance, if L is 12×12 and $b_s = 3$, and if $\text{rank}\{b(2)\} = 2$, the matrices U, V and B obtained after $K = 4$ iterations take the form shown in Fig. 1. In this scenario, the matrices $b(2)$ and $u(3)$ are 2×3 and 12×2, respectively.

The main advantage in using a block-bidiagonalization of fixed size is the simplification of the code. If implemented on an array processor, the iteration with a fixed size is more compatible with parallel implementation. However, in the computation of a given number of Ritz triplets, the use of a fixed block size may generate too many Lanczos vectors and reorthogonalizations, involving an increase in complexity. Subsequently, we use the size reduction procedure rather than the block reorthogonalization on a serial computer.

IV. Adaptive Algorithms with an $O(m^2d)$ Complexity

In electrical engineering applications, we most often require "good" accuracy rather than "full" accuracy. Indeed, in most applications, a relative error of 1e − 4 is sufficient. This decreases the complexity of the iterative algorithms, and makes them more attractive. Adaptive algorithms are iterative algorithms that track parameters of a system which may vary with time, space, or frequency, or more abstractly snapshot. Those systems are called nonstationary, and the accuracy of estimates such as (4) is limited by $\beta(t)$ whose maximal value is function of an equivalent "stationarity duration." In fact, the optimal value of $\beta(t)$ is a tradeoff between the variance of the estimate $R(t)$ and the bias introduced by averaging rank-one matrices $r(t)\,r(t)^H$ that do not have the same expectation in the strict sense. In other words, the error made at the end of the process will be due to modeling errors rather than to the approximations in the calculations.

As pointed out earlier, if variations with time are tracked, the modification at each time step is generally of rank one. However, in other situations, namely block-processing for instance, the modification can be of small norm but still full rank [58]. The algorithms given in this section assume that the new matrix is available at each step, no matter how it has been modified.

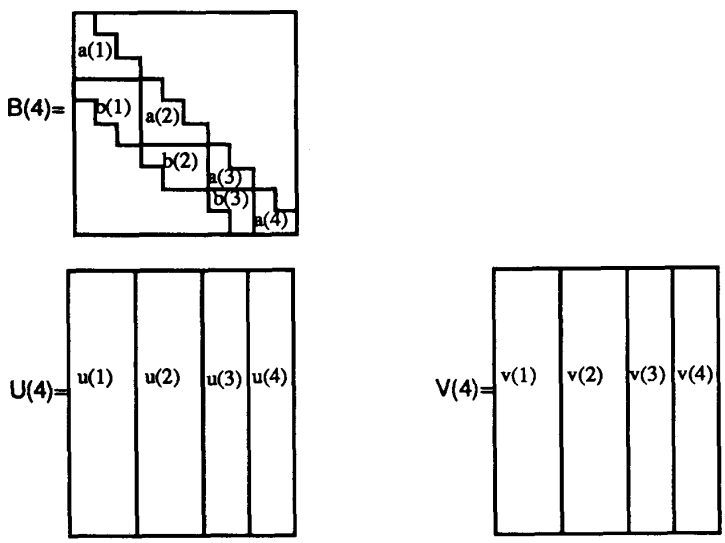


Fig. 1. Form of the matrices generated if L is 12×12. Here, $b_s = 3$, $\text{rank}\{(b(2))\} = 2$, and $K = 4$.

A. Lanczos-based Recursions

In both single-vector and block Lanczos recursions, the loss of orthogonality will occur when the noise level is small or when the sources are almost stationary. In fact, the starting vector (or vectors) may be very close to the range of the singular vectors previously computed, if the matrix varies very slowly.

In the block-Lanczos algorithm, we choose as block size b_s the number d of sources sought (taking $b_s = d + 1$ is also possible and has some advantages). The adaptive algorithm we have selected then takes the form:

Algorithm A, with a block size $b_s = d$ (20)
% Initialization
Start with a narrow $m \times d$ matrix w,
chosen randomly;
% Next time steps **Complexity**
As time increases, $t = 1, \cdots$
Update L according to (5)
Compute the $Kd \times Kd$ matrix
$B(K)$, according to (17) $\boldsymbol{mnKd + 5(m + n)Kd^2/2}$
Compute the left singular pairs of $B(K)$ $\boldsymbol{4K^2d^2}$
Deduce the new left Ritz pairs $\{\theta, w\}$,
according to (12) $\boldsymbol{mK^2d^2}$
Goto $t = t + 1$

In the Algorithm A, K may be chosen in advance and held to a fixed value, in order to limit the number of iterations. If K is never reached, it means that the subspace tracked varies more slowly than expected; if it is often reached, then there is too little time left to achieve the requested accuracy, and K should be increased. If $d = 2 \ll n$, $n = m$ and $K = 2$, the execution of the time loop in the algorithm above has a maximum complexity of order $10n^2 + 24n + O(d^3)$, update of $L(t)$ excluded.

B. Subspace Iteration with Ritz Acceleration (SIR) Running on L(t)

If very few iterations are run, the advantage in using Lanczos recursions vanishes. In these cases, it may be better to use recursions based on the simple power-like method, discarding the search of a tridiagonal form. An important complexity reduction can be done by considering the algorithm below, baptized SIR [29]:

Algorithm B (SIR) (21)
Assume L is $m \times n$ (possibly $m = n$)
% initialization ($t = 0$)
Start with a narrow $m \times d$ matrix w, chosen randomly;
Orthonormalize w; set $U(0) = w$.
Compute $V(0) = L(0)^H * U(0)$;
Orthonormalize $V(0)$.
% Next time steps **Complexity**
As time increases, $t = 1, \cdots$
Update $L(t)$, according to (5) for instance;
% Singular vectors
$W = L(t)^H * U(t - 1)$ $\boldsymbol{nmd/2}$
Compute the $n \times d$ matrix X in the *SVD*:
$X\Sigma Y^H = W$ $\boldsymbol{3nd^2}$
Set $V(t)$ = the d first columns of X.
Compute $W = L(t) * V(t)$; $\boldsymbol{nmd/2}$
Compute the $m \times d$ matrix X in the *SVD*:
$X\Sigma Y^H = W$; $\boldsymbol{3md^2}$
Set $U(t)$ = the d first columns of X.
Goto $t = t + 1$

In this algorithm, note that the singular values are updated twice, and that the Ritz acceleration is still present. Indeed, it consists essentially of running one single iteration of algorithm (14). Now, the number of operations required in the loop (update of $L(t)$ excluded) is $mnd + 3d^2(m + n) + O(md, nd, d^3)$. If $m = n$ and $d = 2 \ll n$, the complexity is roughly $2n^2 + 24n$.

C. Gradient-based Algorithms

In this class can be found the most numerous adaptive algorithms in the literature. For instance, many gradient-based algorithms were developed for the spectral estimation based on Pisarenko harmonic retrieval approach, where the goal is to minimize the objective function $u(t)^H R(t)\, u(t)$ with respect to $u(t)$ under the constraint $u(t)^H u(t) = 1$, in order to estimate the eigenvector associated with the minimal eigenvalue of $\bar{R}(t)$. Among others, one can cite the fixed step stochastic gradient followed by a normalization [59], the stochastic Gauss–Newton method [26], [60], or the conjugate gradient [61]. Somewhat different is the iterative algorithm proposed in [62], where the minimization of the Rayleigh quotient is carried out on the unit sphere appropriately parametrized. Yang and Kaveh review some of the gradient methods applied to the computation of a restricted subset of largest or smallest eigenpairs [63]. Other algorithms utilized in antenna array processing are reviewed by Schreiber in [46]. It is important to distinguish the *exact adaptive gradient* methods and the *instantaneous gradient* methods. In the first technique, the matrix $R(t)$ is requested explicitly, yielding an amount of $O(n^2)$ multiplications [64], [65]; the convergence of the algorithm is controlled by both the averaging coefficients in the updating of $R(t)$ and the coefficient μ. On the other hand, in the instantaneous gradient algorithms the matrix $R(t)$ is replaced by its rank-one update, $r(t)\, r(t)^H$, and can be run at each time step within $O(n)$ multiplications. In this case, the coefficient μ controls the convergence as well as the memory length of the system. The instantaneous gradient algorithm is essentially adaptive, and cannot be used to compute the eigenpairs of a fixed deterministic matrix. We shall present one $O(n^2)$ method in this subsection, and another with $O(n)$ complexity in a subsequent section, in part V. The following algorithm was first proposed by Owsley [64]:

Algorithm C (22)
% initialization ($t = 0$)
Start with a narrow $m \times d$ matrix w, chosen randomly;
Orthonormalize w; set $U(0) = w$.
% Next time steps **Complexity**
As time increases, $t = 1, \cdots$
Update $R(t)$
$g = R(t)\, U(t - 1)$ $\boldsymbol{m^2d}$
$W = U(t - 1) + \mu g$; $\boldsymbol{md}$
% eigenvectors
Orthonormalize W into $U(t)$ $\boldsymbol{2md^2 - 2d^3/3}$
% eigenvalues (optional)
$\Lambda(t) = \mathrm{Diag}(U(t - 1)^H g)$ $\boldsymbol{md}$
Goto $t = t + 1$

Gradient-based adaptive algorithms are unattractive when not used in their approximated instantaneous form; this will be stressed in part V.

D. An Hybrid Method

If the singular triplets vary slowly as t increases, it is possible to decrease the computational load. This can be made for instance by updating each triplet in turn, one at each time step. We call this a "cyclic updating." A cyclic updating strategy was proposed in [27] (with some typos) in order to decrease the complexity by a factor d, and then a slightly different version was published in [66]. We give below the correct square-root version:

Algorithm D1 (23)
Assume L is $m \times n$ (possibly $m = n$)
% initialization ($t = 0$)
Start with a narrow $m \times d$ matrix w, chosen randomly;
Orthonormalize w; set $U(0) = w$.
Compute $V(0) = L(0)^H * U(0)$;
Orthonormalize $V(0)$.
% Next time steps **Complexity**
As time increases, $t = 1, \cdots$
  Update $L(t)$, according to (5) for instance;
  select a value $k \in \{1, 2, \cdots, d\}$
  Compute $w1 = L(t)^H * u_k(t-1)$ **$mn/2$**
  and $w2 = L(t) * v_k(t-1)$ **$mn/2$**
  Orthogonalize $[V(t-1), w1]$ into a
    $n \times (d+1)$ matrix, $\overline{V}$. **$2n(d+1)^2 - 2(d+1)^3/3$**
  Orthogonalize $[U(t-1), w2]$ into a
    $m \times (d+1)$ matrix, $\overline{U}$. **$2m(d+1)^2 - 2(d+1)^3/3$**
  Compute $\overline{B} = \overline{U}^H * L(t) * \overline{V}$ **$mn(d+1)^2$**
  Compute its SVD: $\overline{\Pi} * \overline{\Theta} * \overline{\Gamma}^H = \overline{B}$. **$11(d+1)^3$**
  Retain the d largest singular triplets,
    so that $\overline{B} \approx \Pi * \Theta * \Gamma^H$.
  % Form the Ritz vectors associated with the
    largest d singular values of $L(t)$:
  $U(t) = \overline{U} * \Pi$ and $V(t) = \overline{V} * \Gamma$. **$(m+n)(d+1)^2$**
Goto $t = t + 1$

If $m = n$, and if $d = 2 \ll n$, the execution of the loop in the algorithm above requires $10n^2 + 54n + O(d^3)$ flops, update of $L(t)$ excluded. In [66], the algorithm is not square-root anymore, and the quantities $w1$ and $w2$ are substituted for the difference $w = R(t) * u_k(t-1) - u_k(t-1)\,\lambda(t-1)$, which is then normalized. There are two possibilities, depending on the use of the optional reorthogonalization. The complete adaptive algorithm may be written as:

Algorithm D2 (24)
Assume R is $m \times m$
% initialization ($t = 0$)
Start with a narrow $m \times d$ matrix $\boldsymbol{w}$, chosen randomly;
Orthonormalize $\boldsymbol{w}$; set $U(0) = \boldsymbol{w}$,
Set $\Lambda(0) = I$, the $d \times d$ identity matrix.
% Next time steps **Complexity**
As time increases, $t = 1, \cdots$
  Update $R(t)$
  Select a value $k \in \{1, 2, \cdots, d\}$
  Compute $w = R(t) * u_k(t-1) - u_k(t-1)\,\lambda(t-1)$
  % Option 1: Orthogonalize **$m^2 + m$**
  Normalize $w = w/\|w\|$, **m**
  and form $\overline{U} = [U(t-1), w]$;
  % Option 2: Orthogonalize
  $[U(t-1), w]$ into $\overline{U}$; **$2m(d+1)^2 - 2(d+1)^3/3$**
  Compute $\overline{B} = \overline{U}^H * R(t) * \overline{U}$ **$m^2(d+1)^2$**
  Compute its EVD: $\overline{B} = \overline{\Pi} * \overline{\Theta} * \overline{\Pi}^H$. **$4(d+1)^3/3$**
  Retain the d largest singular triplets,
    so that $\overline{B} \approx \Pi * \Theta * \Pi^H$.
  % Deduce the d left Ritz pairs of $L(t)$:
  $U(t) = \overline{U} * \Pi$; $\Lambda(t) = \Theta$; **$m(d+1)^2$**
Goto $t = t + 1$

This algorithm is like a Lanczos method, and also like a gradient method where the Ritz acceleration would be incorporated; for this reason we called it hybrid. It is now less costly; i.e., if $d = 2 \ll m$, then the execution of instructions inside the loop requires $10m^2 + 28m + O(d^3)$ flops if option 2 is selected. However, one thing is not clear, namely how should we choose the value of k. One can make it cycle in the set $\{1, 2, \cdots, d\}$ as in [27], but in [66], Furhmann seems to let it take a fixed value $k = d$, viz. the value corresponding to the smallest singular triplet that we want to track. In our computer simulations, k is cycling in $\{1, 2\}$ in algorithm D2.

In this part, all the algorithms described had a complexity of order m^2. We may want to reach a complexity of order m, if the processes observed have a sufficiently long stationarity duration. In order to be consistent, it is then necessary to take into account the complexity of the updating of $L(t)$ which is also of order m^2, and to find a way to decrease it to $O(m)$. We study this in the following sections.

V. Adaptive Algorithms with $O(md^2)$ Complexities

In order to decrease the complexity by one order, approximations are now necessary. The goal is to avoid the explicit computation of $L(t)$, as well as matrix-vector products if they are of full size, n. The first algorithm presented (Algorithm E1) expresses the matrix-vector product $R(t)\,U(t-1)$ as a function of $R(t-1)\,U(t-1)$, which is in turn approximated by $U(t-1)\,\Lambda(t-1)$. Algorithm F1 is a square-root and accelerated version of it. It is shown in section F how these approximations can be connected to the estimation of the matrix $R(t)$ by another satisfying a given model; such matrices are said to belong to "class F." A last family of algorithms is then presented, deriving from an approximation in a more accurate class of matrices, called "class G," since it is described in section G. Contrary to the procedures described previously, the algorithms described in this section are fully adaptive, in the sense that they can hardly be used to compute the eigenpairs of a fixed given matrix. In fact, (4) is fully exploited. Moreover, the possible structure or sparsity pattern of the matrix is not taken into account.

E. Approximate Subspace Iteration

The algorithms considered in this section are approximations to Subspace Iteration. In the standard subspace iteration as well as in algorithm SIR, the dominant tasks were (*i*) the updating of $L(t)$ (*ii*) the computation of the matrix-vector products $L(t)^H\,U(t-1)$ and $L(t)\,V(t)$, both of order m^2. It is possible to approximate these two tasks at once [23], [29] and get a solution within $\delta md^2 + O(md + d^2)$ flops, where $\delta = O(1)$.

The first algorithm we describe here was proposed by Karhunen [23], and its convergence in the almost sure sense was proved later in [67]. Here is one version which runs with

$2md^2 + 2md$ flops:

Algorithm E1 (25)
$U(0)$ arbitrary $m \times d$; $U(0)^H U(0) = I$;
$\Lambda(0) = I$, the identity matrix.
For $t = 0, 1, 2, \cdots$ **Complexity**
$z = r(t)^H U(t-1)$; ***md***
$y = \alpha(t)/\beta(t) z \Lambda(t-1)^{-1}$; ***2d***
$w = U(t-1) + r(t) y$; ***md***
$U(t) = \text{ortho}(w)$; $\boldsymbol{2md^2 - 2d^3/3}$
$\lambda_i(t) = \beta(t) \lambda_i(t-1) + \alpha(t) |z_i|^2$, $1 \le i \le d$. ***3d***
Goto $t = t + 1$

B. Modeling of Class F

The algorithm above has the inconvenience that it converges slowly, and does not take advantage of the square-root formulations. This is the reason why improved versions have been studied. it is possible to approximate the SIR technique (22) in the same spirit as for (25). Moreover, the Ritz acceleration can be also maintained. So, the complete algorithm described in [29] may be summarized as follows:

Algorithm F1 (Fast SIR) (26)
$U(0)$ arbitrary $m \times d$; $U(0)^H U(0) = I$;
$\Sigma(0) = I$, the identity matrix
For $t = 0, 1, 2, \cdots$ **Complexity**
$Wu = [\sqrt{\beta}(t)\, \Sigma(t-1);\ \sqrt{\alpha}(t)\, r(t)^H U(t-1)]$; ***2md***
Compute the $(d+1) \times d$ matrix V, in the SVD:
$V \Sigma Y^H = Wu$; $\boldsymbol{6d^3}$
$Wv = [\sqrt{\beta}(t)\, U(t-1)\, \Sigma(t-1);\ \sqrt{\alpha}(t)\, r(t)] V$; $\boldsymbol{md + md^2}$
Compute the $m \times d$ matrix, $U(t)$, in the SVD:
$U(t)\, \Sigma(t)\, Y^H = Wv$; $\boldsymbol{3md^2}$
Goto $t = t + 1$

As pointed out in [29], the singular values are updated twice in F1, but the left singular vectors, $U(t)$, are the same as those of the matrix $[\sqrt{\beta}(t)\, U(t-1)\, \Sigma(t-1),\ \sqrt{\alpha}(t)\, r(t)]$. This suggests another simpler approach. Consider the class of matrices of rank d:

$$\hat{R}^s(t) = U(t) \sum (t)^2 U(t)^H$$

where $U(t)$ is an n by d matrix with orthonormal columns and Σ is diagonal real. If $R(t)$ is replaced by its low-rank approximation, then we have:

$$R(t)\, U(t-1) \approx \hat{R}^s(t)\, U(t-1) = U(t) \sum (t)^2 \qquad (27)$$

We call the approximation $R(t) \approx \hat{R}^s(t)$ "approximation of class F." This shows why the approximations of class F and $R(t-1)\, U(t-1) \approx U(t-1)\, \Lambda(t-1)$ are equivalent if attention is restricted to the subspace iteration. A simpler algorithm can thus be obviously derived from F1:

Algorithm F2 (28)
$U(0)$ arbitrary $m \times d$; $U(0)^H U(0) = I$;
$\Sigma(0) = I$, the identity matrix
For $t = 0, 1, 2, \cdots$ **Complexity**
$W = [\sqrt{\beta}(t)\, U(t-1)\, \Sigma(t-1),\ \sqrt{\alpha}(t)\, r(t)]$; ***md***
Compute the $m \times d$ and $d \times d$ matrices, $U(t)$, $\Sigma(t)$,
in the SVD: $U(t)\, \Sigma(t)\, Y^H = W$; $\boldsymbol{3md^2}$
Goto $t = t + 1$

This trivial algorithm requires an order of only $13m$ multiplications if $d = 2$.

G. Modeling of Class G

Before we go into the details, we give some additional notation. First, the "exact" covariance matrix $\bar{R}(t)$, is unknown and satisfies model (3). Namely, the covariance matrix is the sum of a matrix of rank d, and another known up to a scalar factor, which amounts to consider the identity matrix I:

$$\bar{R}(t) = \bar{R}^s(t) + p^2 I. \qquad (29)$$

Second, the estimated matrix, $R(t)$, is defined by the relation (4), and its Cholesky factor is denoted $L(t)$. It is clear that $R(t)$ does not obey equation (3). This estimate is not utilized here because of the high calculation cost. Now let $\hat{R}(t)$ be a new estimate which satisfies the model:

$$\hat{R}(t) = \hat{R}^s(t) + \hat{p}^2(t) I, \text{ where } \hat{R}^s(t) \text{ is of rank } d. \qquad (30)$$

Karasalo proposed to update $\hat{R}^s(t)$ and $\hat{p}(t)$ by using the EigenValue-eigenvector Decomposition (EVD) of a matrix $R^p(t)$ which is not $R(t)$, but defined recursively by:

$$R^p(t) = \beta(t)\, \hat{R}(t) + \alpha(t)\, r(t)\, r(t)^H. \qquad (31)$$

The goal is to compute $\hat{R}^s(t)$ and $\hat{p}(t)$ so that $\hat{R}(t)$ is the best approximation to $R^p(t)$. Thus, $\hat{R}^s(t)$ corresponds to a rank d approximation of $R^p(t)$, and $\hat{p}^2(t)$ is the arithmetic mean of the $m - d$ remaining eigenvalues of $R^p(t)$. More precisely, consider the spectral decomposition of $R^p(t)$:

$$R^p(t) = \sum_{i=1}^{m} \sigma_i(t)^2\, u_i(t)\, u_i(t)^H.$$

Then, $\hat{R}(t)$ is defined by $\hat{R}^s(t)$ and $\hat{p}^2(t)$ via (30):

$$\hat{p}^2(t) = \frac{1}{m-d} \sum_{i=d+1}^{m} \sigma_i^2(t)$$

$$= \frac{1}{m-d} [\sigma_{d+1}^2(t) + (m-d-1)\, \sigma_{d+2}^2(t)]$$

$$\hat{R}^s(t) = \sum_{i=1}^{d} [\sigma_i^2(t) - \hat{p}^2(t)] u_i u_i^H$$

Interestingly enough, these estimates are optimal in the Maximum Likelihood sense under the Gaussian assumption [28]. Now the product between matrix $R^p(t)$ and a vector would require only $O(md^2)$ operations, but not $O(m^2)$ any more. The algorithm described below uses model (30) and involves only the data matrix in order to improve the condition. This technique was originally proposed by Karasalo in [28].

Algorithm G (32)
% initialization
Start with a narrow $m \times d$ semi-unitary matrix, $U(0)$, chosen at random;
$p(0) = 1$; $\Theta(0) = I$, the $d \times d$ identity matrix.
% Next time steps **Complexity**
As time increases, $t = 1, 2, \cdots$
% splitting of $r(t)$ onto signal and noise spaces,
% as: $r = Uz + \omega c$.
$z(t) = U(t-1)^H * r(t)$; $\omega(t) = r(t) - U(t-1) * z(t)$; ***2md***
$c(t) = \text{norm}(\omega(t))$; $\omega(t) = \omega(t)/c(t)$; ***2m***
Form the $d \times (d+1)$ rectangular matrix:

$$B^P(t) = \begin{pmatrix} \sqrt{\beta(t)}\,\Theta(t-1) & 0 & \sqrt{\alpha(t)}\,z(t) \\ 0 & \sqrt{\beta(t)}\,\hat{p}(t-1) & \sqrt{\alpha(t)}\,c(t) \end{pmatrix}$$

Compute the SVD: $X\Sigma(t)Y^H = B^P(t)$; **$11d^3$**

% X and Y are square unitary of size $d+1$

% and $d+2$ respectively.

Finally compute:

$U(t)$ = the d first columns of $[U(t-1), \omega(t)] * X$; **md^2**

$\Theta(t)$ = the $d \times d$ upper left corner in $\Sigma(t)$; (33)

$$\hat{p}^2(t) = \frac{1}{m-d}[\sigma^2_{d+1}(t) + (m-d-1)\,\beta(t)\,\hat{p}^2(t-1)];$$

Goto $t = t+1$

The matrix $\hat{R}(t)$ is determined by the three relations (33). If necessary, it could be computed explicitly according to:

$$\hat{R}(t) = U(t)\,[\Theta^2(t) - \text{diag}\,\{\hat{p}^2(t)\}]\,U(t)^H + \hat{p}^2(t)I. \quad (34)$$

The complexity required to update the d largest left singular pairs is $md^2 + 2md + 2m + 2d^3 + O(d^2)$, which is very low compared to the performance attained (see next section). For instance, if $d = 2$, the global complexity is of order $10m$.

Lastly, note that the technique of Bunch *et al.* can also be used together with model G, and would provide us with a $O(md^2)$ algorithm, as pointed out by Schreiber [46].

H. Instantaneous Stochastic Gradient Methods

We report here the standard instantaneous stochastic normalized gradient initially proposed by Thompson [49] and so-called Data-Projection Method [63]. Both are programmed in such a way that they estimate the dominant eigenpairs:

Algorithm H1 (35)

Start with a narrow $m \times d$ matrix $\boldsymbol{w}$, chosen randomly;

Orthonormalize $\boldsymbol{w}$; set $U(0) = \boldsymbol{w}$.

% Next time steps **Complexity**

As time increases, $t = 1, \cdots$

$y = U(t-1)^H r(t)$; $g = r(t)y^H$; **$2md$**

$W = U(t-1) + \mu(t)g$; **md**

% eigenvectors

Orthonormalize W into $U(t)$. **$2md^2 \pm O(d^3)$**

% eigenvalues (optional)

$\Lambda(t) = \{\Lambda(t-1) + \mu(t)\,\text{Diag}(yy^H)\}/(1+\mu(t))$ **$2d$**

Goto $t = t+1$

Algorithm H2 (DPM) (36)

Start with a narrow $m \times d$ matrix $\boldsymbol{w}$, chosen randomly;

Orthonormalize $\boldsymbol{w}$; set $U(0) = \boldsymbol{w}$.

% Next time steps **Complexity**

As time increases, $t = 1, \cdots$

$y = U(t-1)^H r(t)$; $g = r(t)y^H$; **$2md$**

$W = U(t-1) + \mu(t)g/\|r(t)\|^2$; **$md + m$**

% eigenvectors

Orthonormalize W into $U(t)$. **$2md^2 \pm O(d^3)$**

% eigenvalues (optional)

$\Lambda(t) = \{\Lambda(t-1) + \mu(t)\,\text{Diag}(yy^H)\}/(1+\mu(t))$ **$2d$**

Goto $t = t+1$

In order to keep some coherence in our comparisons, we chose to set $\mu(t)$ equal to $\alpha(t)/\beta(t)$, because $\mu(t)$ also controls the memory length in this algorithm. Then, one can notice that the updating formula for the eigenvalues has the same form as in the OK algorithm (25), which is consistent. The instantaneous stochastic gradient proposed in [26] avoids the normalization and computes directly the normalized eigenvector. However, this algorithm has been designed for the computation of a single eigenpair, and it seems complicated to take into account the orthonormalization of a set of d vectors instead of a mere normalization of a single vector, in the same fashion. Moreover, both implementations (normalized and direct) have roughly the same complexity and convergence properties, but the explicit orthonormalization has the advantage to insure the orthogonality, which is not guaranteed in a direct recursive formulation such as in [26], due to rounding errors.

Stochastic gradient type algorithms have received much attention because of their simplicity, but are actually numerically unstable, as shown in [68]. This fact has been known for a few years, and stabilized versions (the so-called *leaky LMS*) can be obtained by adding additive white noise to the input in order to excite all the ranges [68], [65]. This refinement is rarely incorporated in adaptive spectral analysis algorithms. In a few words, the updating of a parameter W is rewritten as:

$$W(t) = \eta W(t-1) \pm \mu(t)\,\nabla(t)$$

instead of $W(t) = W(t-1) \pm \mu(t)\,\nabla(t)$, where η is chosen close to one, $\eta < 1$. It is easy to see that the coefficient η indeed introduces a "leak" in the memory of the system. This slight modification suffices to stabilize the stochastic gradient algorithms but the drawback is the introduction of a small bias.

VI. Computer Simulations

Comparisons between some of the algorithms are now described. The situations where it is the easiest to identify tracking and convergence capabilities, are those which have a known sudden change in the parameters to be estimated (this is, so to say, a Heaviside step response). Sudden changes of the dominant eigenvalues while eigenvectors remain fixed show no difference among the adaptive algorithms, since all of them perform roughly the same. On the other hand, if dominant eigenvectors are suddenly rotated 90 degrees, one observes different behaviors. In our set of simulations, two random sources ($d = 2$) are generated by MA processes of order 2 driven by independent Gaussian white noises. If E^i denotes the vector formed by all zeros with a one at the ith position, the observation consists of the following:

$$r(t) = s_1(t)E^1 + s_2(t)E^2 + n(t), \qquad \text{for } t \le t_0 = 10$$

and

$$r(t) = s_1(t)E^3 + s_2(t)E^4 + n(t), \qquad \text{for } t > t_0 = 10.$$

Sources and noise are such that $E\{s_1(t)^2\} = 4$, $E\{s_2(t)^2\} = 1$, and $E\{n(t)\,n(t)^H\} = \sigma^2 I$; the size of observations $r(t)$ is $m = 10$, and $\sigma = 0.1$. The values chosen for coefficients $\alpha(t)$ and $\beta(t)$ were the optimal *a posteriori* values, namely $\alpha(t) = 1/(t - t_0)$ for $t > t_0$, $\alpha(t_0) = 1$, and $\beta(t) = 1 - \alpha(t)$; this corresponds to a rectangular window of growing size starting at $t = t_0$.

The estimate of the 2 dominant singular values provided by each algorithm is shown (figures labeled "a") and can

be compared to the values obtained by the direct computation (algorithm *A*, Fig. 2(a)). For the algorithms computing the eigenvalues, their square root has been plotted. But this is not sufficient as a measure of discrepancy between the estimated and actual signal subspaces. The standard measure of distance between subspaces is the set of principal angles (here there are two), and those are displayed on figures labeled "b," in degrees. By "actual" subspace, it is meant the dominant 2-dimensional eigensubspace of $R(t)$ recomputed directly at each time step. Note that it is different from the subspace spanned by the true source directions $\{E^3, E^4\}$, and represents a fair measure since the error can then reach zero (the angles between the subspace computed and the true source directions cannot be zero because of the presence of additive noise and averaging).

It can be seen in Fig. 2(a) that the covariance matrix attains good approximations to the ideal singular values (which are 1 and 2) at about time 30. This figure shows the precision that cannot be overstepped, since we talk about the *exact* singular values of the *estimated* data matrix available at that time. Subspace Iteration with Ritz acceleration (SIR) does about as good as possible (compare Fig. 2(a) and Fig. 3(a)); in fact Fig. 3(b) shows that after a few steps, both principal angles are smaller than one degree, which indicates convergence.

From Fig. 4, it can be seen that algorithm E has rather poor tracking capabilities because of its relatively slow convergence rate (the first principal angle is always small; the distance must be measured with the help of the second angle, plotted in dotted line). Also, convergence has been proved [67] in the almost sure sense, which is not a very strong topology. Note the singular values after 200 time steps.

Fig. 5(a) and 5(b) demonstrate excellent behavior of algorithm F1. Algorithm G performs as well as F1, but is about half as costly as shown in Table 1. Performances of algorithm F1 are shown in Fig. 7, and are comparable to those of F2. However, it may be stated that algorithm G performs better in certain cases, due to its more general observation model. Note that the scales are different in Figs. 5, 6, and 7. Fuhrmann's algorithm D2 is also quite satisfactory in this simulation, as shown in Fig. 8.

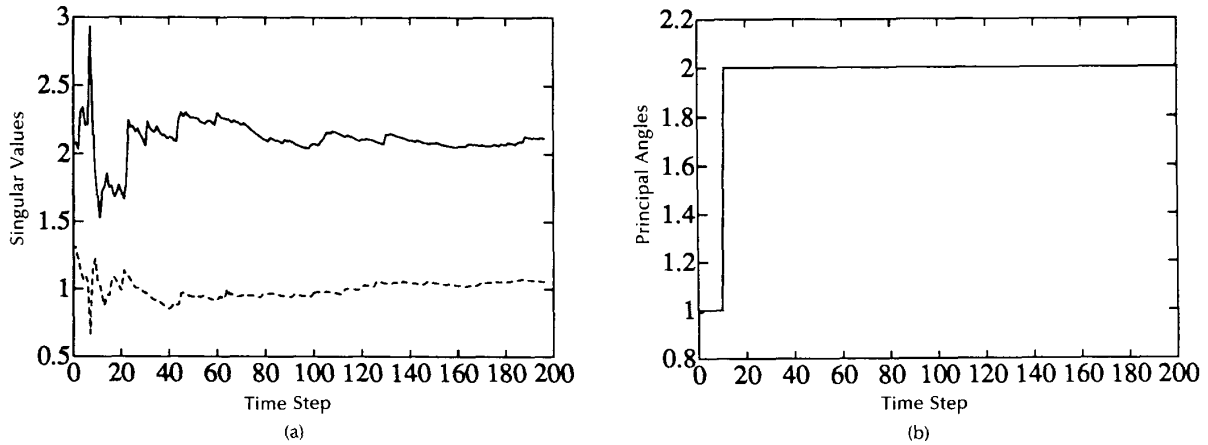


Fig. 2. Algorithm A (Lanczos). (a) Coincides with exact singular values. (b) Witness of sudden change.

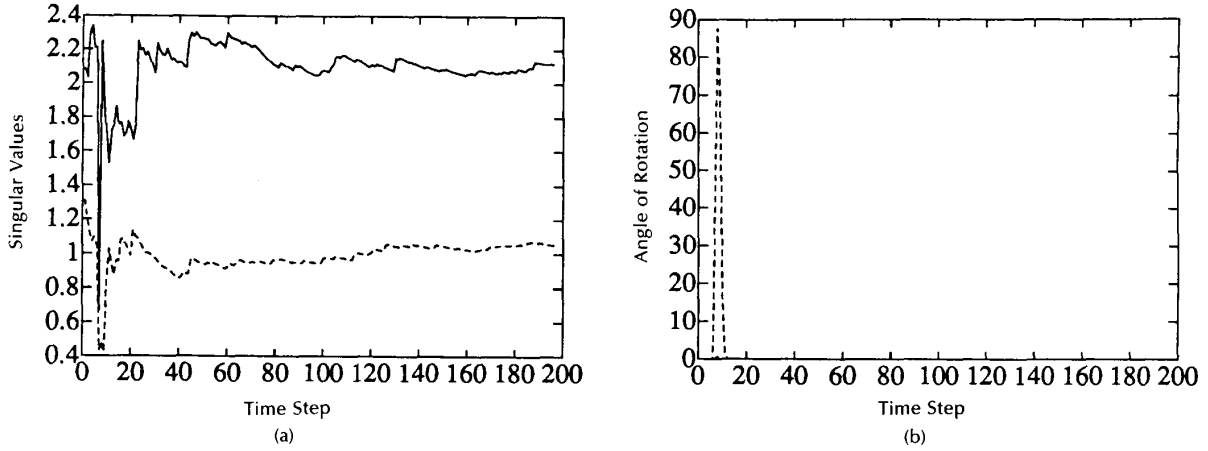
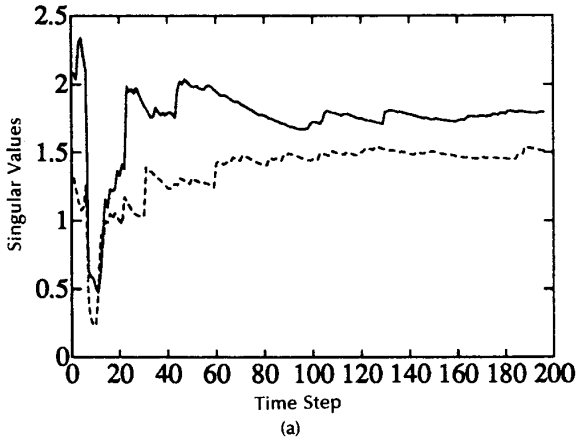


Fig. 3. Algorithm B (SIR). (a) Singular values. (b) Principal angles.



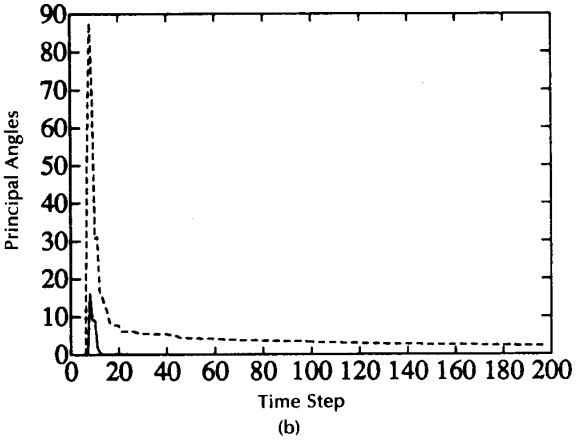
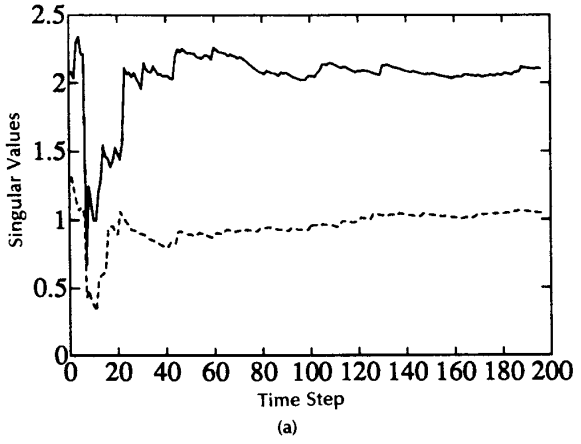


Fig. 4. Algorithm E. (a) Singular values. (b) Principal angles.



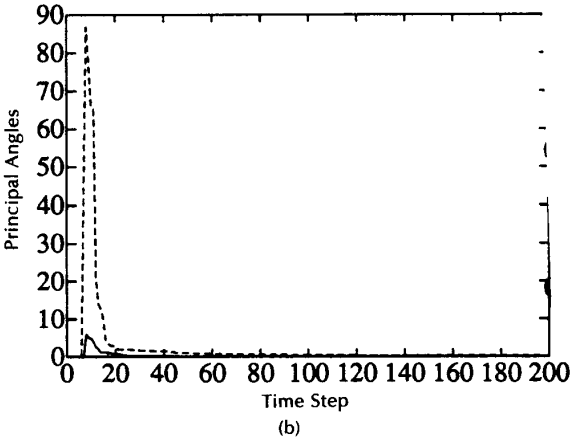


Fig. 5. Algorithm F1. (a) Singular values. (b) Principal angles.

Figures 9 and 10 show the very bad convergence properties of the gradient-based algorithms (one principal angle is stacked at 90 degrees). Indeed, in such block-gradient recursions, the smallest eigenpair computed is always poorly estimated, because of the presence of the noise, among other reasons. Hence, we attempted to examine the improvement brought on by an increase in the block size to $d = 3$. The results obtained are better as seen in Figs. 11 and 12, though they are still much worse than those of algorithms F1, F2, G, and D2. This has been done to the price of an increase in complexity as indicated by the numbers appearing in parentheses in Table 1 (see next section).

VII. Concluding Remarks

Table 1 summarizes the various techniques herein analyzed, and gives an idea of the computational burden involved when 2 sources are sought. We stress that the complexity is not the only criterion that should be considered; besides they differ little from each other. In fact, convergence speed and accuracy are two other important features that are investigated in the previous section. We focused on adaptive algorithms, and for details on more standard ways of computing eigenvalues and eigenvectors, we recommend the reference books [44], [52], or [25].

Algorithm G described in [28] obtains the best performances in our opinion, because of its low cost, and also its excellent convergence and tracking capabilities as well. Moreover, the model utilized to approximate the matrix seems very well matched to the problems arising in signal processing where the observation consists of useful signal, additively corrupted by noise whose covariance matrix is estimated beforehand. As already pointed out, an adaptation of the algorithm of Bunch *et al.* to a matrix modelized as in section V-G is of interest, and has been suggested by Schreiber [46]; this method probably performs very similarly as Karasalo's method (algorithm G). Algorithm F2 is also quite attractive because of its strong regularity and its simplicity. Though it cannot be seen in these simulations, it does not perform as well as algorithm G, and has a slightly higher complexity. We recommend avoiding the use of gradient-based algorithms, which perform poorly and have a comparable complexity.

This study is not exhaustive, and there exist other algo-

Table 1 Listing the Adaptive Algorithms Surveyed*

Section	Part	Reference	Approximate Complexity for $d = 2$, Update of L or R Excluded** $O(m^2)$	$O(m)$	Square-root Y/N	Type of Technique Used
IV						
	A	[0]	**4**	56	Yes	Block-Lanczos with size reduction
	B	[0], [29]	**2**	24	Yes	Subspace iteration + Ritz
	C	[61], [60]	**2**	12	No	Fixed step exact gradient
	D1	[0], [27]	**10**	54	Yes	Gradient + Ritz
	D2	[63]	**10**	28	No	Gradient + Ritz
V						
	E1	[23], [64]		**10**	No	Model F + subspace iteration
	F1	[29]		**21**	Yes	Model F + subspace iteration
	F2	[0], [29]		**13**	Yes	Model F + exact computation
	G	[28]		**10**	Yes	Model G + exact computation
	H1***	[0], [56]		(14) **27**	No	Instantaneous normalized stochastic gradient: (d) $\boldsymbol{d+1}$
	H2***	[60]		(15) **28**	No	Instantaneous normalized stochastic DPM gradient: (d) $\boldsymbol{d+1}$

*[0]means this paper, and corresponds to new contributions.
**The update of matrix L itself, when required according to (5), is not included in the complexity indicated.
***For algorithms H1 and H2, the complexity to consider (in boldface) corresponds to a size $d + 1 = 3$, since the versions of size d perform poorly (complexity in parentheses).

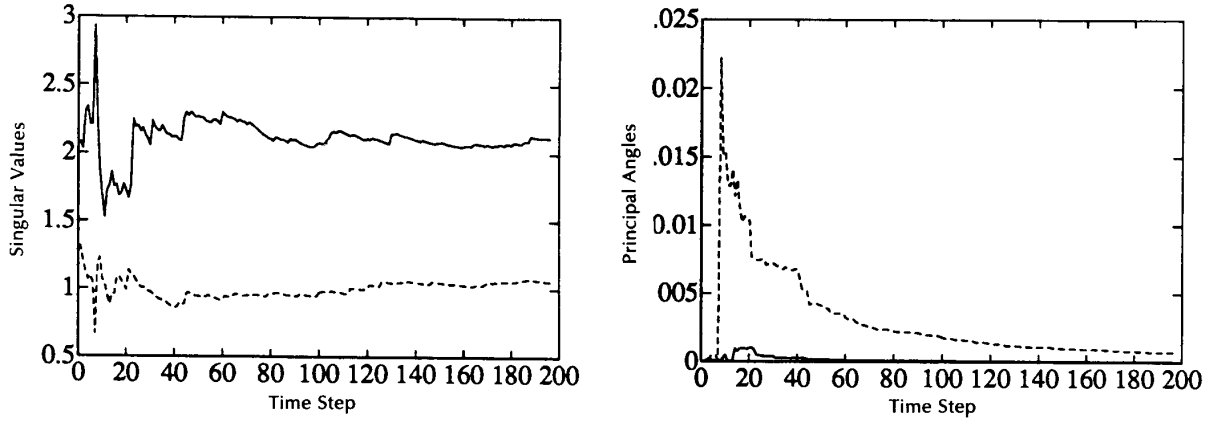


Fig. 6. Algorithm G. (a) Singular values. (b) Principal angles.

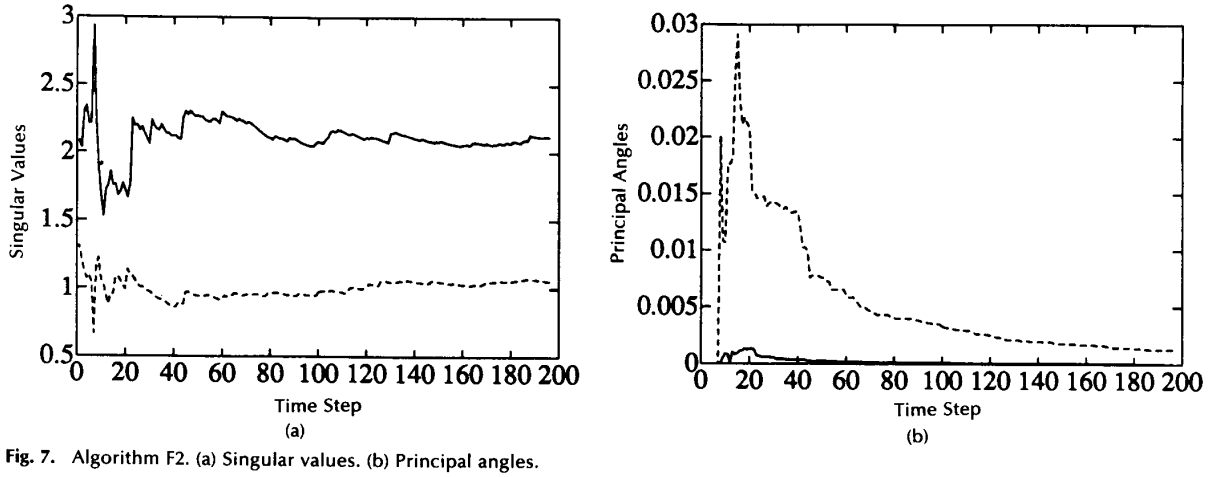


Fig. 7. Algorithm F2. (a) Singular values. (b) Principal angles.

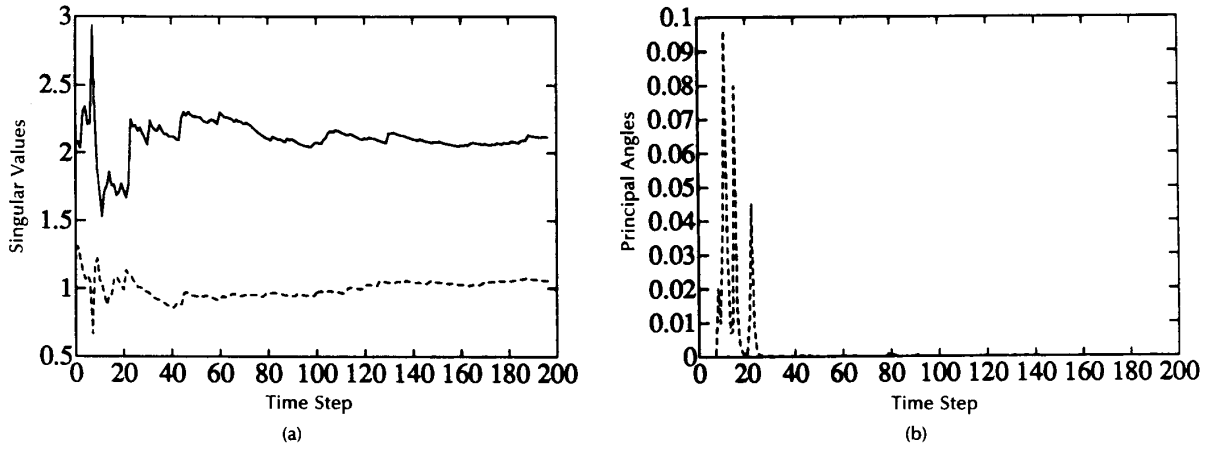


Fig. 8. Algorithm D2. (a) Singular values. (b) Principal angles.

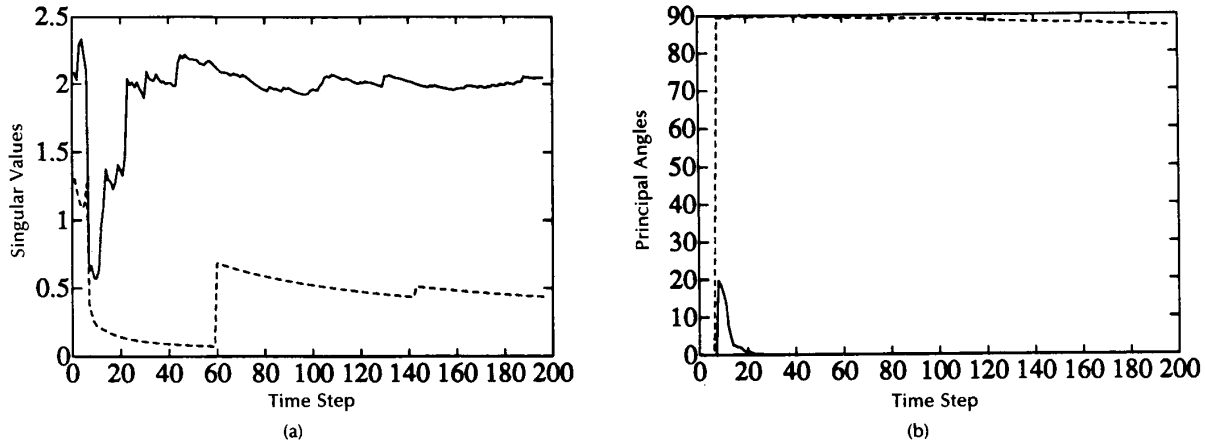


Fig. 9. Algorithm H1, size *d*. (a) Singular values. (b) Principal angles.

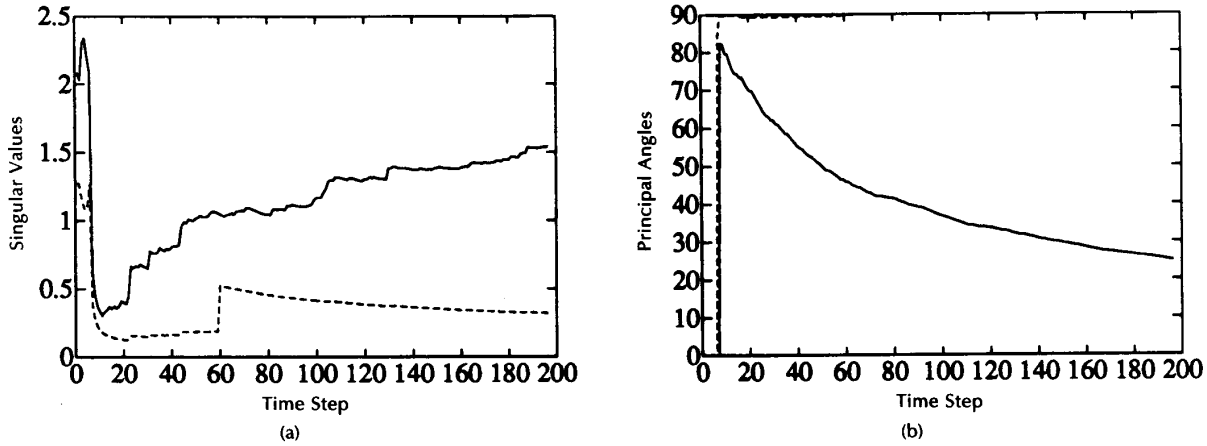


Fig. 10. Algorithm H2, size *d*. (a) Singular values. (b) Principal angles.

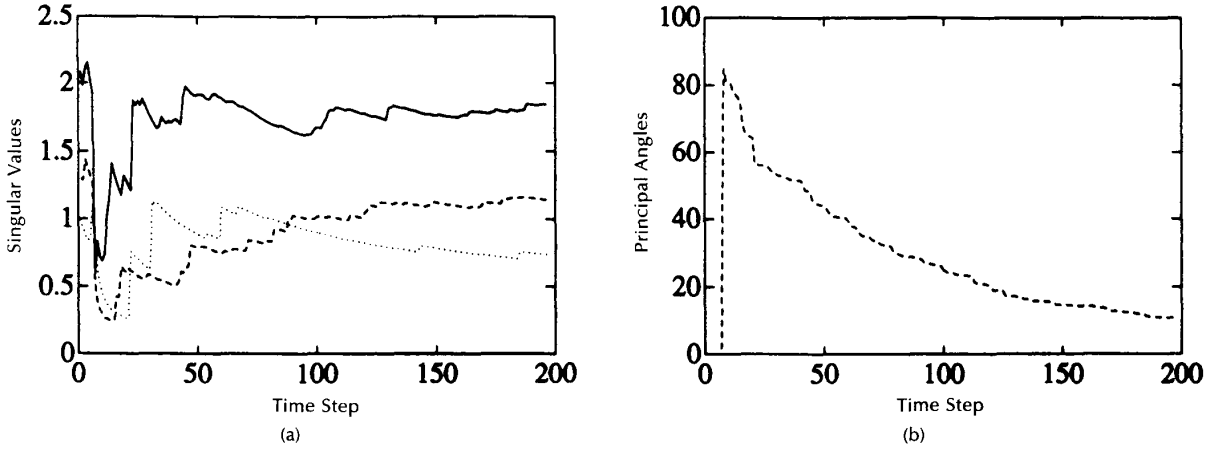


Fig. 11. Algorithm H1, size $d + 1$. (a) Singular values. (b) Principal angles.

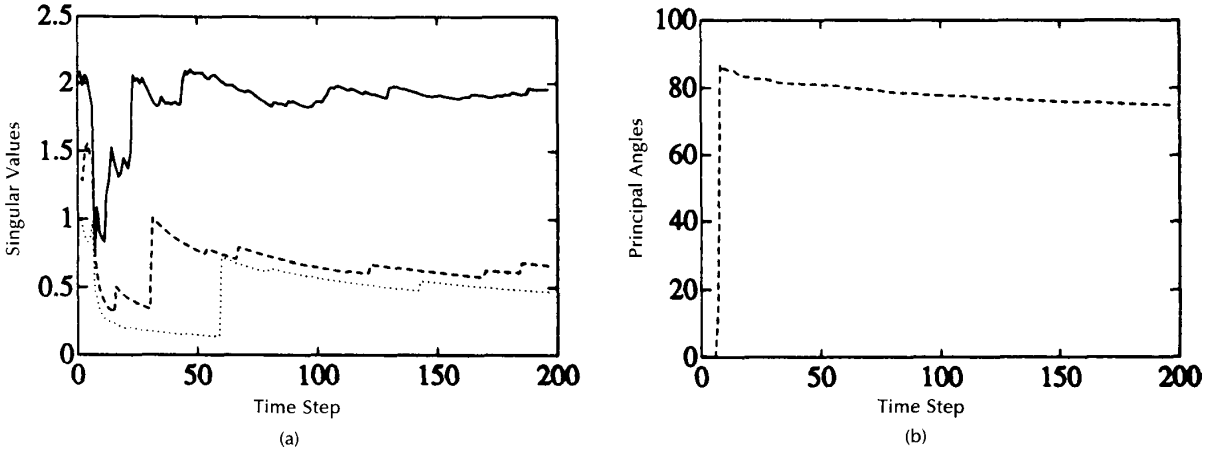


Fig. 12. Algorithm H2, size $d + 1$. (a) Singular values. (b) Principal angles.

Table 2 Complexity of the QR Factorization

Computation of:	Overall Complexity, $m \geq n$	Method Used
Q in factored form & R	$mn^2 - n^3/3$	Householder
Q accumulated & R	$2m^2n - mn^2 + n^3/3$	Householder
Q_1 accumulated & R	$2mn^2 - 2n^3/3$	Householder
Q_1 accumulated & R	mn^2	Gram-Schmidt

rithms, such as the constrained subspace approximation (CSA) introduced by Hu [60], or the instantaneous stochastic Gauss-Newton iteration introduced by Reddy *et al.* [26], that could probably be adapted to the computation of a restricted subset of dominant eigenpairs. We have restricted ourselves to a limited number of algorithms for the sake of clarity and brevity, but this does not mean that our choice has been made only in favor of the best ones. The Generalized EigenValue Problem (GEVD) or the GSVD has been skipped in the same spirit, but they remain of interest, and are mostly unsolved with respect to adaptive implementations. The solution of whitening the data is still the solution used in practice [46], though it could be performed in another manner, at least by a Lanczos-type approach.

Appendix

In this appendix, we recall the complexity of the QR and SVD factorizations, as a function of the matrix size. For further details, the reader is invited to consult [25, pp. 148–152] for QR, and [32, p. 11.18] and [25, p. 175] for the SVD.

Complexity of Standard QR Factorizations

Let A be any m by n matrix, and assume without restricting the generality that $m \geq n$. Then there exist a $m \times m$ unitary matrix, Q, and a $n \times m$ upper triangular matrix R, such that

$$A = Q[R^T \quad 0]^T = Q_1 R.$$

Table 3 Complexity of the Singular Value Decomposition

Computation of:	Golub-Reinsch Algorithm		Chan Algorithm
	Complexity with $m = n$	Complexity with $m > n$	Complexity with $m \gg n$
Σ	$4n^3/3$	$2mn^2 - 2n^3/3$	mn^2
Σ & U	$6n^3$	$2m^2n + 4mn^2$	$2m^2n$
Σ & U_1	*	$7mn^2 - n^3$	$3mn^2$
Σ & U & V	$11n^3$	$2m^2n + 4mn^2 + 14n^3/3$	$2m^2n$
Σ & U_1 & V	*	$7mn^2 + 11n^3/3$	$3mn^2$

In this expression, Q_1 is defined as the m first columns of Q. Therefore, matrix Q_1 satisfies $Q_1^H Q_1 = I$. A being given, table 2 gives the complexity of the computation of these matrices.

Complexity of Standard Singular Value Decompositions

Let A be any $m \times n$ matrix, and assume without restricting the generality that $m \geq n$. Then there exist two unitary matrices, U and V, of size $m \times m$ and $n \times n$, respectively, and a $n \times n$ real diagonal matrix Σ such that:

$$A = U \begin{bmatrix} \Sigma \\ 0 \end{bmatrix}^T V^H = U_1 \Sigma V^H.$$

In this expression, U_1 is defined as the n columns of U associated with nonzero singular values, and is therefore a $m \times n$ matrix satisfying $U_1^H U_1 = I$. Table 3 gives the complexity of the computation of these various matrices. The rightmost column in Table 3 refers to an algorithm described in [69].

References

[1] K. V. Mardia *et al.*, *Multivariate Analysis*. New York, NY: Academic Press, 1979.

[2] W. K. Pratt, *Digital Image Processing*. New York, NY: Wiley, 1975.

[3] B. C. Moore, "Principle component analysis in linear systems, controllability, observability, and Model Reduction," *IEEE Trans on Automatic Control*, vol TAC-26, no. 1, pp. 17–32, Feb. 1981.

[4] H. L. Van Trees, *Detection, Estimation and Modulation theory*, vol. 1. New York, NY: Wiley, 1968.

[5] Y. T. Chien and K. S. Fu, "On the generalized Karhunen-Loève expansion," *IEEE Trans. Inform. Theory*, vol. TI-13, no. 3, pp. 518–520, 1967.

[6] S. Y. Kung and K. S. Arun, "SVD algorithms for linear system approximation and spectrum estimation," in *Advances in Statistical Signal Processing*, vol. 1, JAI Press Inc., 1987, pp. 203–250.

[7] T. Kailath, *Linear Systems. Englewood Cliffs, NJ: Prentice Hall, 1980.*

[8] R. Courant and D. Hilbert, *Methods of Mathematical Physics*. New York, NY: Wiley, 1953.

[9] J. C. Samson, "Pure states, polarized waves and principal components in the spectra of multiple geophysical time series," *Geophys. Jour. Roy. Astr. Soc.*, vol. 72, no. 3, pp. 647–664, 1983.

[10] F. Glangeaud and C. Latombe, "Identification of electromagnetic sources," *Annales Geophysicae*, vol. 1, no. 3, pp. 245–252, 1983.

[11] V. F. Pisarenko, "The retrieval of harmonics from a covariance function," *Geophys. Jour. Roy. Astr. Soc.*, vol. 33, pp. 347–366, 1973.

[12] T. N. Ulrych and R.W. Clayton, "Time series modeling and maximum entropy," *Physics of the Earth and Planeary Interiors*, vol. 12, pp. 188–199, 1976.

[13] S. Haykin, Ed. *Nonlinear Methods of Spectral Analysis*. New York, NY: Springer-Verlag, 1983.

[14] G. Bienvenu and L. Kopp, "Optimality of high-resolution array processing using the eigensystem approach," *IEEE Trans. ASSP-31*, no. 5, pp. 1235–1248, Oct. 1983.

[15] R. O. Schmidt, "Multiple emitter location and signal parameter estimation," *Proc. RADC Spectrum Estimation Workshop*, NY 1979, reprinted in *IEEE Trans. on Antenna and Propag.*, vol. AP-34, no. 3, pp. 276–280, Mar. 1986.

[16] O. C. Zienkiewicz, *The Finite Element Method*. New York, NY: McGraw Hill, 1977.

[17] R. Roy, A. Paulraj and T. Kailath, "Esprit, a subspace rotation approach to estimation of parameters of cisoids in noise," *IEEE Trans. on ASSP*, vol. ASSP-34, no. 5, pp. 1340–1342, Oct. 1986.

[18] P. Comon, "An array processing technique using the first principal component," in *SVD and Signal Processing*, M. Deprettere, Ed. Amsterdam, The Netherlands: North Holland, 1988.

[19] R. Roy, "ESPRIT—Estimation of Signal Paramaters via Rotational Invariance Techniques," Ph.D. Thesis, Stanford University, Aug. 1987.

[20] R. Roy, and T. Kailath, "ESRIT—Estimation of Signal Parameters via Rotational Invariance Techniques," *IEEE Trans. ASSP*, vol. 37, no. 7, pp. 984–995, July 1989.

[21] K. Aziz and A. Settari, *Petroleum Reservoir Simulation*. Applied Science Publishers, 1979.

[22] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms*, vol. 1. Birkhauser, 1985.

[23] J. Karhunen, "Adaptive algorithm for estimating eigenvectors of correlation-type matrices," in *Proc. ICASSP*, San Diego, 1984, pp. 14.6.1–4.

[24] D. Pal, A. Ligtenberg and T. Kailath, "An $O(n^2 \log n)$ method for finding the eigenvalues of a symmetric Toepliz and Hankel matrices," preprint, Jan 1987.

[25] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Hopkins, 1983.

[26] V. U. Reddy, B. E. Gardt and T. Kailath, "Least squares type algorithm for adaptive implementation of Pisarenko's harmonic retrieval method," *IEEE ASSP*, vol. 30, no. 3, pp. 399–405, 1982.

[27] D. R. Fuhrmann, "Adaptive MUSIC," in *Proc. SPIE Conf.*, San Diego, 1987.

[28] I. Karasalo, "Estimating the covariance matrix by signal subspace averaging," *IEEE Trans. on ASSP*, vol. 34, no. 1, pp. 8–12, Feb. 1986.

[29] P. Comon, "Fast computation of a restricted subset of eigenpairs of a varying hermitian matrix," in *Proc. NATO ASI on Numerical Linear Algebra, Signal Processing and Parallel Processing*, Aug. 1–12, 1988, Leuven, Belgium.

[30] G. H. Golub, "Some modified matrix eigenvalue problems," *SIAM Review*, vol. 15, no. 2, pp. 318–334, Apr. 1973.

[31] J. R. Bunch, C. P. Nielsen, "Updating the singular value decomposition," *Numerische Math.*, vol. 31, pp. 111–129, 1978.

[32] J. J. Dongarra *et al.*, *Linpack User's Guide*. SIAM, 1979.

[33] B. S. Garbow *et al.*, *EISPACK, Lecture Notes in Computer Science*, vol. 51. New York, NY: Springer-Verlag, 1977.

[34] C. Moler, S. Herskovitz *et al.*, *Matlab User's Guide*. S. Natick, MA: The Mathworks Inc., 1989.

[35] J. P. Lecadre, "Parametric methods for spatial signal processing in the presence of unknown colored noise fields," to appear in *IEEE Trans. ASSP*.

[36] H. Hung and M. Kaveh, "Focusing matrices for coherent signal-subspace processing," *IEEE Trans. ASSP*, pp. 1272–1281, Aug. 1988.

[37] P. E. Gill, G. H. Golub, W. Murray, and M. A. Saunders, "Methods for modifying matrix factorizations," *Math. Comp.*, vol. 28, no. 126, pp. 505–535, Apr. 1974.

[38] G. J. Bierman and C. Thornton, "Numerical comparison of Kalman filter algorithms: orbit determination case study," *Automatica*, vol. 13, pp. 23-35, 1977.
[39] T. Kailath, "Alternatives in discrete-time recursive estimation," *Int. Jour. Control*, 1980, vol. 32, no. 2, pp. 311-328. Also reprinted in *Lectures on Wiener and Kalman Filtering*, T. Kailath, Ed. New York, NY: Springer-Verlag, 1981.
[40] M. Verhaegen and P. Van Dooren, "Numerical aspects of different Kalman filter implementations," *IEEE Trans. on AC*, vol. 31, no. 10, pp. 907-917, Oct. 1986.
[41] J. M. Cioffi, A Fast QR frequency-domain RLS adaptive filter," *Proc. ICASSP*, Dallas, Apr. 1987, vol. 1, pp. 407-410.
[42] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1986.
[43] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Math*, vol. 14, pp. 403-420, 1970, or *Handbook of Automatic Computation*, Wilkinson and Reinsch, Ed. New York, NY: Springer-Verlag, 1971.
[44] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. Oxford, UK:Clarendon Press, 1965.
[45] R. J. Vaccaro *et al.*, "Advances in principal component signal processing," *First Workshop on SVD and Signal Processing*, M. Deprettere Ed. Amsterdam, The Netherlands: North Holland.
[45b] G. Peters and J. H. Wilkinson, "The calculation of specified eigenvectors by inverse iteration," in *Linear Algebra. Handbook for Automatic Computation*, J. H. Wilkinson and C. Reinsch, Eds. New York, NY: Springer-Verlag, 1971.
[46] R. Schreiber, "Implementation of adaptive array algorithms," *IEEE Trans. ASSP*, vol. 34, no. 5, pp. 1038-1045, Oct. 1986.
[47] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank one modification of symmetric eigenproblem," *Numerische Math.*, vol. 31, pp. 31-48, 1978.
[48] C. Davis and W. Kahan, "The rotation of eigenvectors by a perturbation," *SIAM Jour. Num. Anal.*, vol. 7, pp. 1-46, 1970.
[49] R. C. Thompson, "The behavior of eigenvalues and singular values under perturbations of restricted rank," *Linear Algebra and its Applications*, vol. 13, pp. 69-78, vol. 13, 1976.
[50] B. N. Parlett, "Remarks on matrix eigenvalue computations," in *VLSI and Modern Signal Processing*, S. Y. Kung, H. J. Whitehouse, and T. Kailath Eds. Englewood Cliffs, NJ: Prentice-Hall, 1985.
[51] C. C. Paige, "Accuracy and effectiveness of the Lanczos algorithm for the symmetric eigenproblem," *Linear Algebra and its Applications*, vol. 34, pp. 235-258, 1980.
[52] B. N. Parlett, *The Symmetric Eigenvalue Problem*. Englewood Cliffs, NJ: Prentice-Hall, 1980.
[53] B. N. Parlett, H. Simon, and L. M. Stringer, "On estimating the largest eigenvalue with Lanczos algorithm," *Mathematics of Computation*, vol. 36, no. 157, pp. 153-165, Jan. 1982.
[54] P. Comon, "Adaptive computation of a few extreme eigenpairs of a positive definite hermitian matrix," *Proc. of EUSIPCO*, Sept. 5-8, 1988, Grenoble, France.
[55] G. H. Golub and W. Kahan, "Calculating the singular values and pseudo inverse of a matrix," *SIAM Journal Num. Anal.*, ser. B, vol. 2, no. 2, pp. 205-224, 1965.
[56] C. C. Paige, "Bidiagonalization of matrices and solution of linear equations," *SIAM Jour. Numer. Analysis*, vol. 11, no. 1, Mar. 1974.
[57] G. H. Golub, F. T. Luk, and M. L. Overton, "A block lanczos method for computing the singular values and corresponding singular vectors of a matrix," *ACM Trans. Math. Soft.*, pp. 149-169, June 1981.
[58] P. Comon and D. Trystram, "An incomplete factorization algorithm for adpative filtering," *Signal Processing*, vol. 13, no. 4, pp. 353-360, Dec. 1987.
[59] M. G. Larimore, "Adaptation convergence of spectral estimation based on Pisarenko harmonic retrieval," *IEEE Trans. ASSP*, pp. 955-962, Aug. 1983.
[60] Y. H. Hu, "Adaptive methods for real time Pisarenko spectrum estimate. in *Proc. ICASSP 85*, 1985, pp. 105-108.
[61] H. Chen and T. K. Sarkar *et al.*, "Adaptive spectral estimation by the conjugate gradient method," *IEEE Trans. ASSP*, vol. 34, no. 2, pp. 272-284, 1986.
[62] D. R. Fuhrmann and B. Liu, "An iterative algorithm for locating the minimal eigenvector of a symmetric matrix," in *Proc. ICASSP 84*, 1984, pp. 45.8.1-4.
[63] J. F. Yang and M. Kaveh, "Adaptive eigensubspace algorithms for direction or frequency estimation and tracking," *IEEE Trans. ASSP*, vol. 36, no. 2, pp. 241-251, Feb. 1988.
[64] N. L. Owsley, "Adaptive data orthogonalization," in *Proc. Conf. ICASSP*, 1978, pp. 109-112.
[65] B. Windrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
[66] D. R. Fuhrmann, "An algorithm for subspace computation with applications in signal processing," *SIAM Jour. Matrix Anal. Appl.*, vol. 9, no. 2, Apr. 1988.
[67] E. Oja and J. Karhunen, "On stochastic approximation of the eigenvectors and eigenvalues of the expectation of a random matrix," *Journal of Math Analysis Appl.*, vol. 106, no. 1, pp. 69-84, Feb. 1985.
[68] J. M. Cioffi, "Limited precision effects in adaptive filtering," *IEEE Trans. CAS*, pp. 821-833, July 1987.
[69] T. F. Chan, "An improved algorithm for computing the SVD," *ACM Trans. Math. Soft.*, vol. 8, pp. 84-88.

Pierre Comon (Member, IEEE) was born in Paris, France, on January 28, 1959. He received the doctorate degree in electrical engineering in 1985 from the University of Grenoble, France.

From 1982 to 1985 he was supported by Crouzet Ltd, Valence, France, while he was with the Center of Geophysics and Random Phenomena Studies (CEPHAG), France, from 1981 to 1986. In 1987 he visited the Information Systems Laboratory at Stanford. He has been with the Department of Submarine Activities, Thomson-Sintra, France, since February 1988. His current interests include signal theory, parallel algorithms, sonar array processing, and neural networks. He may be reached by e-mail as na.comon@na-net.stanford.edu.

Dr. Comon is a member of Eurasip. INNS, and SIAM.

Gene H. Golub (Honorary Member, IEEE) was born in Chicago on Feb. 29, 1932. He was educated in the Chicago public school system. He attended the University of Illinois where he received his doctorate in 1959. Subsequently, he held an NSF Fellowship at the University of Cambridge.

He joined the faculty of Stanford University in 1962, where he is currently a Professor of Computer Science at Stanford and the Director of the newly established program in Scientific Computing and Computational Mathematics. He served as chairman of the CS department from 1981 until 1984. He is noted for his work in the use of numerical methods in linear algebra for solving scientific and engineering problems. This work has resulted in a book, *Matrix Computations*, co-authored with Charles Van Loan. He has served as President of the Society of Industrial and Applied Mathematics (1985-87). He is the founding editor of two SIAM journals: *The SIAM Journal of Scientific and Statistical Computing* and *The SIAM Journal of Matrix Analysis and Applications*. He is the originator of na-net and can be reached by e-mail as na.golub@na-net.stanford.edu.

Dr. Golub holds four honorary degrees, is a member of The National Academy of Engineering and is a Fellow of the AAAS.