Fast Algorithms for Updating Signal Subspaces

Guanghan Xu, *Member, IEEE*, Hongyuan Zha, *Member, IEEE*, Gene Golub, *Honorary Member, IEEE*, and Thomas Kailath, *Fellow, IEEE*

***Abstract*—In various real-time signal processing and communication applications, it is often required to track a low-dimensional signal subspace that slowly varies with time. Conventional methods of updating the signal subspace rely on eigendecomposition or singular value decomposition, which is computationally expensive and difficult to implement in parallel. Recently, Xu and Kailath proposed fast and parallelizable Lanczos-based algorithms for estimating the signal subspace based on the data matrices or the covariance matrices. In this paper, we shall extend these algorithms to achieve fast tracking of the signal subspace. The computational complexity of the new methods is $O(M^2d)$ per update, where M is the size of the data vectors and d is the dimension of the signal subspace. Unlike most tracking methods that assume d is fixed and/or known *a priori*, the new methods also update the signal subspace dimension. More importantly, under certain stationarity conditions, we can show that the Lanczos-based methods are *asymptotically equivalent* to the more costly SVD or eigendecomposition based methods and that the estimation of d is *strongly consistent*. Knowledge of the previous signal subspace estimate is incorporated to achieve better numerical properties for the current signal subspace estimate. Numerical simulations for some signal scenarios are also presented.**

I. Introduction

IN THE AREA of detection and estimation, there is a class of high-performance algorithms, termed *signal subspace* algorithms, that has various applications in direction finding and signal waveform recovery, mobile communications, spectral estimation, system identification, just to name a few. These algorithms all incorporate the common key step of estimating the signal subspace, termed *signal subspace decomposition*, which is conventionally achieved by eigendecomposition (ED) or the singular value decomposition (SVD). For real-time implementation, we need to update the signal subspace or equivalently a small number of extreme eigenvectors or singular vectors for every batch of incoming data vectors. Updating ED or SVD has been a long-standing and extensively-studied problem in numerical linear algebra [1]–[3, ch. 8]. In [4], Comon and Golub gave a detailed review of various methods for tracking a few extreme singular values and vectors. In particular, two classes of methods are proposed with computational complexities $O(M^2d)$ and $O(Md^2)$ flops, where M is the size of the data vectors and d is the dimension of the signal subspace. The basic procedure of the first class ($O(M^2d)$) is to update the QR factorization and then to apply various methods (e.g., Lanczos algorithm, subspace iteration algorithm) for estimating the required extreme singular values and vectors [5]–[9]. In this context, another class of methods based on the so-called URV and ULV decompositions has also been developed by Stewart and *et al.* [10]–[12]. Though these techniques require more computation, they are more stable since the error accumulation is not significant. The second class (i.e., $O(Md^2)$) of algorithms exploits the underlying structure of the ideal covariance matrices (i.e., low-rank matrix + shift) to achieve an order of magnitude of complexity reduction [13]. More numerically robust versions of the above techniques have been proposed by DeGroat and Roberts [14], [15]. These algorithms can be useful in the scenarios where the covariance matrix can be very well approximated by the *ideal* one. However, in many real applications, the covariance matrix usually does not match the ideal model so well. As a consequence, this class of algorithms will involve an *approximation* step and the whole procedure will suffer potential instability due to error accumulation.

In almost all the existing methods mentioned above, the estimation errors and convergence properties were not analyzed. More importantly, most of the above mentioned techniques assume the signal subspace dimension d to be *known* and *fixed a priori*. In many applications, e.g., direction finding and mobile communications, some signals can be off and on from time to time; and tracking the number of signals present can be of paramount importance. We should mention that Griffin and Stewart [11], [12] also considered the problem of tracking the signal subspace dimension and proposed an interesting scheme based on the rank-revealing URV (ULV) decomposition.

In this paper, we shall extend some recently developed Lanczos-based algorithms [16], [17] to the problem of tracking signal subspaces, which includes different detection schemes for estimating the signal subspace dimension. In fact, the proposed methods are similar to some approaches in [4], although the approaches in [4] never considered the detection problem. If the signal subspace at each data block remains the same (i.e., the scenario is stationary), we can also show that the new approach is *asymptotically equivalent* to the computationally more costly ED or SVD based methods and that the detection schemes we develop are *strongly consistent.*

Manuscript received November 11, 1991; revised February 3, 1993. G. Xu and T. Kailath were supported in part by the Joint Services Program at Stanford University (US Army, US Navy, US Air Force) under Contract DAAL03-88-C-0011, by the Army Research Office under Contract DAAL03-89-K-0109, and by grants from the General Electric Company and Boeing ARGO Systems Inc. H. Zha and G. Golub were supported in part by the Army Research Office under Contract DAAL03-90-G-0105. This paper was recommended by Associate Editor M. A. Soderstrand.

G. Xu is with the Dept. of Electrical & Computer Engineering, The University of Texas at Austin, Austin, Texas 78712-1084 USA.

H. Zha is with the Dept. of Computer Sciences, Pennsylvania State University, University Park, PA 16802 USA.

G. Golub is with the Dept. of Computer Sciences, Stanford University, Stanford, CA 94305 USA.

T. Kailath is with the Dept. of Computer Sciences & Information Systems Lab., Stanford University, Stanford, CA 94305 USA.

IEEE Log Number 9402402.

Based on some new results on the convergence of the Lanczos algorithm, we shall also show how to determine a good starting vector for the current update based on knowledge of the signal subspace at the previous step. Such a starting vector enables well-balanced convergence of the Lanczos algorithm and thus avoids potential numerical instability. In this paper, we also give a more numerically stable alternative to the Lanczos algorithm based on the use of Jacobi rotations.

This paper is organized as follows. In the next section, we define the problem. In Section III, we focus on procedures for QR updating and downdating. In Section IV, we describe the Lanczos-based algorithm and some detection schemes for determining the number of signals. We also present the above mentioned stabler alternative to the Lanczos algorithm using the Jacobi rotations. A novel way of selecting a good starting vector, and a brief performance analysis of the Lanczos-based algorithms, are also described. Results of numerical simulations on some examples are reported in Section V.

II. Problem Statement

A. Data Model

Many signal estimation and detection problems can be formulated as parameter estimation problems based on noisy measurements. The measured sample vectors ($M \times 1$) can be represented as

$$\mathbf{x}(t) = \mathbf{A}(t)\mathbf{s}(t) + \mathbf{n}(t) \tag{1}$$

where $\mathbf{s}(t) \in \mathcal{C}^{d\times 1}$ denotes the signal vector and $\mathbf{n}(t) \in \mathcal{C}^{M\times 1}$ the noise vector at time t; $\mathbf{A}(t)$ is an $M\times d$ matrix containing d basis vectors that span the so-called *signal subspace* $\mathcal{S}(t)$. In a large class of so-called signal subspace algorithms, the first step is to determine the signal subspace $\mathcal{S}(t)$, based on which the other signal parameters can be estimated. In most applications, this is the most computationally intensive part. In this paper, the signal subspace is assumed to be time-varying and we shall show how to update the signal subspace when new data vectors are added.

B. Ideal Covariance Matrices

If we assume that the signal and noise are uncorrelated, then the data covariance matrix can be expressed as

$$\mathbf{C_x}(t) = \mathbf{A}(t)\mathbf{C_s}(t)\mathbf{A}^H(t) + \mathbf{C_n}(t) \tag{2}$$

where $\mathbf{C_x}(t)$, $\mathbf{C_s}(t)$, and $\mathbf{C_n}(t)$ represent the data covariance matrix, the signal covariance matrix, and the noise covariance matrix respectively [17]. The noise vectors can often be assumed to be white[1], i.e., $\mathbf{C_n}(t) = \sigma_n^2(t)\mathbf{I}$. Thus

$$\mathbf{C_x}(t) = \mathbf{A}(t)\mathbf{C_s}(t)\mathbf{A}^H(t) + \sigma_n^2(t)\mathbf{I}. \tag{3}$$

[1]If the noise is correlated, we will assume that the normalized noise covariance $\mathbf{\Sigma_n(t)}$ (but not necessarily the power $\sigma_n^2(t)$ is either known or can be estimated; therefore, the data covariance matrix $\mathbf{C_x}(t)$ can be *pre-whitened.*

C. Sample Covariance Matrices

In many applications, though the signal and noise vectors change rather rapidly, their statistics vary slowly with time. Therefore as an approximation, we may assume that the covariance $\mathbf{C_x}(t)$ remains the same for a certain period of time, say for N samples. Let us write it in terms of its eigendecomposition

$$\mathbf{C_x}(t) = \sum_{k=1}^{M} \lambda_k \mathbf{e}_k \mathbf{e}_k^H \tag{4}$$

where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$ are the eigenvalues and the $\{\mathbf{e}_k\}$ are the corresponding eigenvectors of $\mathbf{C_x}$. From (3), it follows that $\lambda_{d+1} = \cdots = \lambda_M = \sigma_n^2$. Defining $\mathbf{E_s} = [\mathbf{e}_1, \mathbf{e}_2, \cdots, \mathbf{e}_d]$ and $\mathbf{E_n} = [\mathbf{e}_{d+1}, \mathbf{e}_{d+2}, \cdots, \mathbf{e}_M]$, we can show that $\mathbf{E_s}$, i.e., the set of eigenvectors corresponding to the d largest eigenvalues, spans the *signal subspace*, i.e.,

$$span\{\mathbf{E_s}\} = span\{\mathbf{A}(t)\}. \tag{5}$$

Clearly, the signal subspace dimension d can be determined from the multiplicity of the noise eigenvalue, if d is not known *a priori*.

In reality, however, the ideal covariance $\mathbf{C_x}(t)$ is not available. We only have an estimate ($\hat{\mathbf{C}}_\mathbf{x}(t)$) of $\mathbf{C_x}(t)$ based on a finite number (N) of snapshots

$$\hat{\mathbf{C}}_\mathbf{x} = \frac{1}{N}\sum_{t=1}^{N} \mathbf{x}(t)\mathbf{x}^H(t). \tag{6}$$

If ($\hat{\lambda}_k$, $\hat{\mathbf{e}}_k$), $1 \leq k \leq M$, are eigenpairs of $\hat{\mathbf{C}}_\mathbf{x}$, $\hat{\lambda}_{d+1}, \cdots, \hat{\lambda}_M$ will not be exactly the same and $span\{\hat{\mathbf{e}}_1, \cdots, \hat{\mathbf{e}}_d\} \neq span\{\mathbf{A}(t)\}$. However, for reasonably large N and slowly varying $\mathbf{C_x}(t)$, $\hat{\lambda}_{d+1}, \cdots, \hat{\lambda}_M$ are confined to a small region of size $O(1/\sqrt{N})$ [18], [19] and $span\{\hat{\mathbf{e}}_1, \cdots, \hat{\mathbf{e}}_d\} \approx span\{\mathbf{A}(t)\}$. Thus one robust way of tracking the signal subspace is to compute a complete eigendecomposition of $\hat{\mathbf{C}}_\mathbf{x}$ and then estimate the number of sources based on the eigenvalue distribution. In fact, there are many well-known detection schemes available such as the likelihood ratio, MDL, and AIC tests [20]. The basic idea of these detection schemes is to find $\hat{d}$ by checking whether $\hat{\lambda}_{\hat{d}+1}, \cdots, \hat{\lambda}_M$ are close to one another or equivalently by checking whether their arithmetic and geometric means are close to each other. According to [18], the following statistic

$$L(\hat{d}) = -N(M-\hat{d})\log\left\{\frac{\left(\prod_{k=\hat{d}+1}^{M}\hat{\lambda}_k\right)^{1/(M-\hat{d})}}{\frac{1}{M-\hat{d}}\sum_{k=\hat{d}+1}^{M}\hat{\lambda}_k}\right\} \tag{7}$$

is a likelihood ratio, with the following property:

$$L(\hat{d}) = \begin{cases} O(N) & \hat{d} < d \\ O(1) \overset{N\to\infty}{\sim} \mathcal{X}^2((M-\hat{d})^2 - 1) & \hat{d} = d \end{cases} \tag{8}$$

where $\mathcal{X}^2(p)$ denotes the χ-square distribution with degrees of freedom p. The dramatic change in the behavior of $L(\hat{d})$ when $\hat{d} = d$ is the crucial fact underlying detection schemes. Knowing the limiting distribution of $L(\hat{d})$ when $\hat{d} = d$, we can determine a threshold and compare $L(\hat{d})$ with it in order to estimate d; this gives us the sequential likelihood ratio test. We can also rely on more objective information theoretic criteria, i.e., AIC and MDL criteria:

$$AIC(\hat{d}) = L(\hat{d}) + \hat{d}(2M - \hat{d}) \tag{9}$$

$$MDL(\hat{d}) = L(\hat{d}) + \frac{1}{2}\hat{d}(2M - \hat{d})\log N. \tag{10}$$

The minimizers of the AIC and MDL criteria are chosen as the AIC and MDL estimates of the d. According to [20], MDL is strongly consistent, i.e., the probability of correct detection approaches one as $N \to \infty$. Once the detection is carried out (i.e., d is determined), the signal subspace is spanned by the first d eigenvectors of the sample covariance matrix. However, computing the eigendecomposition involves at least $O(M^3)$ flops for an $M \times M$ covariance matrix and it is also difficult to be implemented on parallel machines. Recently, Xu and Kailath developed a fast subspace decomposition approach [17], [22] using the Lanczos algorithm. This new approach exploits the structure of the sample covariance matrices (i.e., they have $M - d$ closely-clustered eigenvalues) and only requires $O(M^2 d)$ flops to implement. Since in most applications, $d \ll M$, the Lanczos-based algorithm gives almost an order of magnitude of computational reduction. In the following, we shall apply this new approach for updating the slowly varying signal subspaces.

III. Updating Cholesky Factorization

Although formation of the sample covariance matrix $\hat{\mathbf{C}}_{\mathbf{x}}(t)$ costs $O(NM^2)$ flops, it only requires $O(M^2 n)$ to recursively update $\hat{\mathbf{C}}_{\mathbf{x}}(t)$, where n is the number of *new* data vectors. In fact, we only need to perform an updating/downdating n times so that:

$$\begin{aligned}\hat{\mathbf{C}}_{\mathbf{x}}(t) &= \frac{1}{N}\sum_{\tau=t-N+1}^{N} \mathbf{x}(\tau)\mathbf{x}^H(\tau)\\ &= \hat{\mathbf{C}}(t-n) + \frac{1}{N}\sum_{\tau=t-n+1}^{t} \mathbf{x}(\tau)\mathbf{x}^H(\tau)\\ &\quad - \frac{1}{N}\sum_{\tau=t-N-n+1}^{t-N} \mathbf{x}(\tau)\mathbf{x}^H(\tau).\end{aligned} \tag{11}$$

Once $\hat{\mathbf{C}}_{\mathbf{x}}$ is computed, the batch Lanczos-based algorithm can be readily applied to estimate the signal subspace, as well as its dimension. In fact, this is quite a straightforward approach to updating the signal subspace. However, as is pointed in [23], the covariance update is not numerically accurate. In real-time implementation, the loss of precision stems from the procedure of squaring the data matrix (more precisely, forming the cross-product). In this section, we shall focus on another numerically more accurate way of updating the signal subspace by directly working on the data matrix. Instead of updating the covariance matrix, we propose to update the square root (Cholesky factor) of the covariance matrix or the upper-triangular matrix $\mathbf{R}$ in the QR factorization of the data matrix $\mathbf{X}_N(t)$ $(N \times M)$. We shall briefly describe the updating procedure, *well-known* to numerical analysts for the benefit of readers less familiar with numerical computations.

Suppose that we have completed the QR factorization of the data matrix $\mathbf{X}_N(t)$ containing N data vectors (see (12) at the bottom of the page) where $\mathbf{Q} = [\mathbf{q}_1 \ \mathbf{q}_2 \ \cdots \ \mathbf{q}_M]$ contains M orthonormal vectors and $\mathbf{R}$ is an upper-triangular matrix. The estimated signal subspace is spanned by the eigenvectors corresponding to the d largest eigenvalues. Now the problem is to update the matrix $\mathbf{R}$ when the data matrix is augmented by a new data vector $\mathbf{x}(t+1)$, which can be done in many ways. In the following, we show one simple QR updating method [3]. The idea is that we first increase the dimension of $\mathbf{Q}$ and $\mathbf{R}$ by one as shown below. Now consider the matrix

$$\mathbf{X}_{N+1}(t+1) = \begin{bmatrix} \mathbf{X}_N(t) \\ \frac{1}{\sqrt{N}}\mathbf{x}^H(t+1) \end{bmatrix}. \tag{13}$$

Then

$$\begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix}\begin{bmatrix} \mathbf{R} \\ \frac{1}{\sqrt{N}}\mathbf{x}^H(t+1) \end{bmatrix} = \mathbf{Q}'\mathbf{R}'. \tag{14}$$

Now we can zero out the last row of $\mathbf{R}'$ via a sequence of Jacobi rotations, i.e.,

$$\mathbf{R}' = \mathbf{P}\begin{bmatrix} \tilde{\mathbf{R}} \\ \mathbf{0} \end{bmatrix} \tag{15}$$

where $\mathbf{P}$ is the product of the Jacobi rotations and therefore is orthogonal, then

$$\mathbf{X}_{N+1}(t+1) = \mathbf{Q}\mathbf{P}\begin{bmatrix} \tilde{\mathbf{R}} \\ \mathbf{0} \end{bmatrix} = \tilde{\mathbf{Q}}\tilde{\mathbf{R}} \tag{16}$$

where $\tilde{\mathbf{Q}}$ is the first M columns of $\mathbf{QP}$, and $\tilde{\mathbf{R}}$ gives the updated upper-triangular factor.

To zero out the last row of $\mathbf{R}'$ in (14) requires M Jacobi rotations as shown below:

$$\begin{array}{c} \rightarrow \\ \\ \\ \\ \rightarrow \end{array}\begin{bmatrix} \times & \times & \cdots & \times \\ & \times & \cdots & \times \\ & & \ddots & \vdots \\ & & & \times \\ \times & \times & \cdots & \times \end{bmatrix} \Longrightarrow \begin{array}{c} \\ \rightarrow \\ \\ \\ \rightarrow \end{array}\begin{bmatrix} \times & \times & \cdots & \times \\ & \times & \cdots & \times \\ & & \ddots & \vdots \\ & & & \times \\ 0 & \times & \cdots & \times \end{bmatrix} \Longrightarrow \bullet\bullet\bullet$$

$$\Longrightarrow \begin{array}{c} \\ \\ \\ \rightarrow \\ \rightarrow \end{array}\begin{bmatrix} \times & \times & \cdots & \times \\ & \times & \cdots & \times \\ & & \ddots & \vdots \\ & & & \times \\ 0 & 0 & \cdots & \times \end{bmatrix} \Longrightarrow \begin{bmatrix} \times & \times & \cdots & \times \\ & \times & \cdots & \times \\ & & \ddots & \vdots \\ & & & \times \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

$$\mathbf{X}_N(t) = \frac{1}{\sqrt{N}}\begin{bmatrix} \mathbf{x}^H(t-N+1) \\ \mathbf{x}^H(t-N+2) \\ \vdots \\ \mathbf{x}^H(t) \end{bmatrix} = [\mathbf{q}_1 \ \mathbf{q}_2 \ \cdots \ \mathbf{q}_M]\begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1M} \\ & r_{22} & \cdots & r_{2M} \\ & & \ddots & \vdots \\ & & & r_{MM} \end{bmatrix} = \mathbf{QR}. \tag{12}$$

Since we are only interested in updating the matrix $\mathbf{R}$, there is no need for storing and updating the $\mathbf{Q}$ matrix. Therefore, the computational cost of the updating procedure is $O(M^2)$ flops. For simplicity of presentation, we count a multiplication as one flop in this paper.

To compute $\mathbf{X}_N(t+1)$, we also need to downdate $\mathbf{R}$ by deleting an old data vector $\frac{1}{\sqrt{N}}\mathbf{x}(t-N+1)$ from $\mathbf{X}_{N+1}(t+1)$. The procedure is the same as the above updating one except that Jacobi rotations for nulling out the last row of $\mathbf{R}'$ should be replaced by hyperbolic rotations. The hyperbolic rotation matrix $\mathbf{H}$ used for this purpose is a rank-two modification of the identity matrix $\mathbf{I}$ and it satisfies $\mathbf{H}^H\mathbf{J}\mathbf{H} = \mathbf{I}$ where $\mathbf{J} = diag\{1, 1, \cdots, 1, -1\}$ [3], [24}.

If it is only required to modify the signal subspace for every n snapshots, then the above procedure can be repeated n times, resulting in a computational complexity of $O(M^2 n)$ flops, which is of the same order as in the covariance updating procedure (11).

IV. Fast Estimation of Signal Subspace

A. A Review of the Bidiagonalization Lanczos Algorithm

Given the upper-triangular matrix $\mathbf{R}$, the signal subspace is spanned by its first d right singular vectors. A complete SVD of $\mathbf{R}$ requires $O(M^3)$ flops. In [4], a bidiagonalization Lanczos algorithm (Bi-Lanczos) is applied to estimate the d principal singular values and vectors.

The Bi-Lanczos Algorithm *[3]:*

```
Given $\mathbf{R}$ ($M \times M$); $\mathbf{p}_0 = \mathbf{v}_1$ (unit-norm); $\beta_0 = 1$;
$j = 0$; $\mathbf{u}_0 = 0$
while $\beta_j \neq 0$
    $\mathbf{v}_{j+1} = \mathbf{p}_j/\beta_j$; $j := j + 1$
    $\mathbf{r}_j = \mathbf{R}\mathbf{v}_j - \beta_{j-1}\mathbf{u}_{j-1}$; $\alpha_j = \|\mathbf{r}_j\|$; $\mathbf{u}_j = \mathbf{r}_j/\alpha_j$
    $\mathbf{p}_j = \mathbf{R}^H\mathbf{u}_j - \alpha_j\mathbf{v}_j$; $\beta_j = \|\mathbf{p}_j\|$
end
```

The above algorithm can also be written in a more concise form [3], for $j = 1, 2, \ldots$

$$\mathbf{R}\mathbf{v}_j = \alpha_j\mathbf{u}_j + \beta_{j-1}\mathbf{u}_{j-1} \qquad \beta_0\mathbf{u}_0 \equiv 0 \tag{17}$$

$$\mathbf{R}^H\mathbf{u}_j = \alpha_j\mathbf{v}_j + \beta_j\mathbf{v}_{j+1} \qquad \beta_M\mathbf{v}_{M+1} \equiv 0. \tag{18}$$

At each step, say the jth step, if we denote $\mathbf{U}_j = [\mathbf{u}_1, \cdots, \mathbf{u}_j]$ and $\mathbf{V}_j = [\mathbf{v}_1, \cdots, \mathbf{v}_j]$, then it can be shown that

$$\mathbf{U}_j^H\mathbf{R}\mathbf{V}_j = \mathbf{B}_j = \begin{bmatrix} \alpha_1 & \beta_1 & & \cdots & 0 \\ 0 & \alpha_2 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & \beta_{j-1} \\ 0 & \cdots & & 0 & \alpha_j \end{bmatrix}. \tag{19}$$

Let $\{\mathbf{a}_i^{(j)}, \theta_i^{(j)}, \mathbf{b}_i^{(j)}\}_{i=1}^j$ be the singular value triplets of $\mathbf{B}_j$, where $\theta_i^{(j)}$ is the ith singular value and $\mathbf{a}_i^{(j)}$ and $\mathbf{b}_i^{(j)}$ are the corresponding left and right singular vectors. Then for $j \geq d$, (i.e., after more than d Lanczos steps), $\{\theta_i^{(j)}\}_{i=1}^d$, termed Rayleigh-Ritz (RR) values are used to approximate the d principal singular values of $\mathbf{R}$ or $\mathbf{X}_N(t+1)$ and $\{\mathbf{y}_i\}_{i=1}^d = \{\mathbf{V}_j\mathbf{b}_i\}_{i=1}^d$, termed as RR vectors, are estimates of the d principal right singular vectors of $\mathbf{R}$ or $\mathbf{X}_N(t+1)$. The process using the RR values and vectors to approximate the singular values and vectors is called *Rayleigh–Ritz (RR) approximation.* Since $\mathbf{B}_j$ is a small $(j \times j)$ bidiagonal matrix and computation of singular values alone only requires $9j^2$ flops and computation of $\{\sigma_i, \mathbf{v}_j\}_{i=1}^j$ takes $O(j^3)$ flops.

B. Convergence of the Rayleigh–Ritz Approximation

From the above algorithm, it is easily seen that the major computation at each Lanczos step is two matrix-vector products $\mathbf{R}\mathbf{v}_j$ and $\mathbf{R}^H\mathbf{u}_j$. In [4], based on some results in [3] on error bounds for the RR approximation, it was claimed that estimation error using the RR values and vectors becomes negligible if α_j or β_j and other related quantities become very small. However, there was no rigorous analysis on how fast the α_j or β_j converges to zero and more importantly the analysis of the estimation error of the RR approximation was not given, although it was observed that with only $j = O(d)$ iterations the estimation error of the RR vectors can be made negligible and thus the whole computational complexity of one update is $O(M^2 j) = O(M^2 d)$. In the following, we shall prove that this claim is true; and moreover, if the signal subspace is assumed to be fixed for these N data samples, we can also show that the signal subspace estimate based on the RR vectors at the $d+2$ steps is *asymptotically equivalent* to that obtained from a more costly SVD based method. Thus, $O(M^2 d)$ flops are sufficient to compute an estimate of the signal subspace.

As we know, a *sample* covariance matrix $\hat{\mathbf{C}}_\mathbf{x}(t) \triangleq \mathbf{X}_N^H(t)\mathbf{X}_N(t)$ is an $O(N^{-1/2})$ estimate of $\mathbf{C}_\mathbf{x}$, i.e.,

$$\mathbf{C}_\mathbf{x}(t) - \hat{\mathbf{C}}_\mathbf{x}(t) = O(N^{-1/2}). \tag{20}$$

Let $\{\sigma_i^2, \mathbf{v}_i\}_{i=1}^M$ denote the eigenvalues and eigenvectors of $\mathbf{C}_\mathbf{x}(t)$ and $\{\hat{\sigma}_i, \hat{\mathbf{v}}_i\}_{i=1}^M$ the singular values and right singular vectors of $\mathbf{X}_N(t)$.[2] (see [18])

$$\mathbf{v}_k - \hat{\mathbf{v}}_k = O(N^{-1/2}), \qquad k = 1, 2, \cdots, d, \tag{21}$$

$$\sigma_k - \hat{\sigma}_k = O(N^{-1/2}), \qquad k = 1, 2, \cdots, M. \tag{22}$$

Therefore,

$$\begin{aligned} &dist(span\{\mathbf{A}(t)\}, span\{\hat{\mathbf{v}}_k\}) \\ &\quad = dist(span\{\mathbf{v}_k\}_{k=1}^d, span\{\hat{\mathbf{v}}_k\}_{k=1}^d) \\ &\quad = O(N^{-1/2}) \end{aligned} \tag{23}$$

where $dist(\cdot,\cdot)$ denotes the subspace distance, which is defined in [3, p. 76.r]. Equation (23) means that the d principal right singular vectors of $\mathbf{X}_N(t)$ span an estimated subspace that is only $O(N^{-1/2})$ away from the true signal subspace.

It has been shown in [22], [17] that the RR values and vectors have the following important asymptotic properties: for $j > d$

$$\theta_k^{(j)} - \hat{\sigma}_k = O(N^{-(j-d)}), \qquad k = 1, 2, \cdots d, \tag{24}$$

$$\mathbf{y}_k^{(j)} - \hat{\mathbf{v}}_k = O(N^{-(j-d)/2}), \qquad k = 1, 2, \cdots, d. \tag{25}$$

[2] It is crucial to note that in (21) and (22), as well as in the following text, the symbols $O(\cdot)$, $o(\cdot)$ denote "in probability" versions of the corresponding usual deterministic notations, unless they are specified otherwise. For the rigorous definition of these notations, see [21], Section 2.9.

Hence, once $m \geq d+2$,

$$\lim_{N\to\infty} \sqrt{N}(\mathbf{y}_k^{(m)} - \mathbf{v}_k) = \lim_{N\to\infty} \sqrt{N}(\hat{\mathbf{v}}_k - \mathbf{v}_k), \qquad k = 1, 2, \cdots, d. \quad (26)$$

In other words, *span*$\{\mathbf{y}_k^{(m)}\}_{k=1}^d$ and *span*$\{\hat{\mathbf{v}}_k\}_{k=1}^d$ are *asymptotically equivalent* to the estimates of the *true* signal subspace *span*$\{\mathbf{A}(t)\}$ = *span*$\{\mathbf{v}_k\}_{k=1}^d$.

It is worth noting that (24) and (25) are only valid for the first d RR values. For $k \geq d+1$, according to [22], we can only show that $\theta_k^{(j)} - \hat{\sigma}_k = O(N^{-1/2})$. This may be related to the fact that the Lanczos algorithm is good at extracting the extreme singular values but not the near-repeated ones.

C. Estimation of the Signal Subspace Dimension

From (12), it follows that $\hat{\mathbf{C}}_\mathbf{X}(t) = \mathbf{X}_N^H(t)\mathbf{X}_N(t) = \mathbf{R}^H\mathbf{R}$. it is easily seen that the squares of the singular values of $\mathbf{X}_N(t)$ or $\mathbf{R}$, i.e., $\{\hat{\sigma}_i^2\}_{i=1}^M$, are eigenvalues of $\hat{\mathbf{C}}_\mathbf{X}(t)$. Replacing $\hat{\lambda}_i$ by $\hat{\sigma}_i^2$ in (7), (9), and (10), gives rise to various detection schemes, e.g., AIC and MDL, for estimating the signal subspace dimension d. The fundamental problem that still remains with the Bi-Lanczos algorithm for detecting d is that not all of the singular values $\hat{\sigma}_i$ are available at each intermediate step. Instead, we only have a few RR values, which could be *asymptotically equivalent* estimates of the corresponding principal singular values of $\mathbf{R}$. However, by (7), the likelihood ratio relies on the product and quadratic sum of the smaller squared singular values of $\mathbf{R}$, i.e., $\prod_{i=\hat{d}}^M \hat{\sigma}_i^2$ and $\sum_{i=\hat{d}}^M \hat{\sigma}_i^2$, which we do not seem to know at any intermediate Bi-Lanczos step.

The trick to get around this difficulty is to evaluate the quadratic sum and product of all the singular values, which are the squares of the Frobenius norm and the determinant of $\mathbf{R}$, respectively, i.e.,

$$\|\mathbf{R}\|_F^2 = \sum_{i=1}^M \hat{\sigma}_i^2, \qquad det(\mathbf{R})^2 = \prod_{i=1}^M \hat{\sigma}_i^2. \quad (27)$$

With good estimates of some of the larger singular values, we can subtract or divide these two quantities and the *desired* product and quadratic sum of smaller singular values will follow. Thus,

$$\prod_{k=\hat{d}+1}^M \hat{\sigma}_k^2 = det(\mathbf{R})^2 / \prod_{k=1}^{\hat{d}} \hat{\sigma}_k^2 \approx det(\mathbf{R})^2 / \prod_{k=1}^{\hat{d}} \theta_k^{(j)^2}, \quad (28)$$

$$\sum_{k=\hat{d}+1}^M \hat{\sigma}_k^2 = \|\mathbf{R}\|^2 - \sum_{k=1}^{\hat{d}} \hat{\sigma}_k^2 \approx \|\mathbf{R}\|^2 - \sum_{k=1}^{\hat{d}} \theta_k^{(j)^2} \quad (29)$$

and $L(\hat{d})$ is estimated by $\varphi(\hat{d})$:

$$\varphi(\hat{d}) = -N(M-\hat{d}) \log \left\{ \frac{\left(det(\mathbf{R})^2 / \prod_{k=1}^{\hat{d}} \theta_k^{(j)^2} \right)^{1/(M-\hat{d})}}{\frac{1}{M-\hat{d}} \left(\|\mathbf{R}\|_F^2 - \sum_{k=1}^{\hat{d}} \theta_k^{(j)^2} \right)} \right\}. \quad (30)$$

Clearly, the computational complexity for $\|\mathbf{R}\|_F$ is $\frac{1}{2}M(M+1) \sim O(M^2)$ flops, which is marginal in comparison with the total computational cost. It usually requires $O(M^3)$ flops to compute $det(\mathbf{R})$. Now, since $\mathbf{R}$ is a upper-triangular matrix $\{r_{ij}\}$, $det(\mathbf{R}) = \prod_{i=1}^M r_{ii}$ and the computational cost becomes $O(M)$.

If $j \geq d+2$, $\theta_k^{(j)} = \hat{\sigma}_k + O(N^{-(j-d)}) = \hat{\sigma}_k + o(N^{-1})$ and therefore the difference of $\varphi(\hat{d})$ and $L(\hat{d})$ is $o(1)$. Hence, by (8), it is not difficult to see that $\varphi(\hat{d})$ also has the following interesting property:

$$\varphi(\hat{d}) = \begin{cases} O(N) & \hat{d} < d \\ O(1) \overset{N\to\infty}{\sim} \mathcal{X}^2((M-\hat{d})^2 - 1) & \hat{d} = d \end{cases} \quad (31)$$

Now, if $j \leq d$, then $\theta_k^{(j)}$, $1 \leq k \leq j$, do not converge to the corresponding singular values with probability one, thus $\varphi(\hat{d}) = O(N)$ for all $\hat{d} \leq j$. However, if $j = d+1$, (30) still holds but the distribution of $\varphi(d)$ is unknown. The trick to get around this dilemma is that we only evaluate $\varphi(\hat{d})$ for $\hat{d} = 1, \cdots, j-2$ at the jth Lanczos step. In this case, the property (30) of $\varphi(\hat{d})$ always holds.

With $\varphi(\hat{d})$, we can develop a new detection scheme similar to the likelihood ratio test. The basic idea is that we set a confidence level α say 99% or 99.5%, based on which a threshold $\gamma_{\hat{d}}$ can be determined; at each step, $\varphi(\hat{d})$ and $\gamma_{\hat{d}}$ are compared and the first $\hat{d}$ such that $\varphi(\hat{d}) < \gamma_{\hat{d}}$ is chosen to be the estimate of d. The detailed procedure is listed below.

The FSD Detection Scheme

1) Set $\hat{d} = 0$.
2) Set null hypothesis $\mathrm{H}_0 : d = \hat{d}$.
3) Choose the threshold $\gamma_{\hat{d}}$ based on the proper $\mathcal{X}^2$ distribution and evaluate $\varphi(\hat{d})$.
4) If $\varphi(\hat{d}) \leq \gamma_{\hat{d}}$ accept H_0 and stop.
5) If $\varphi(\hat{d}) > \gamma_{\hat{d}}$, reject H_0; if $\hat{d} < m-2$, $\hat{d} := \hat{d}+1$, return to 2). Otherwise continue with the Bi-Lanczos algorithm.

Since $\varphi(\hat{d}) = O(N)$ whenever $\hat{d} < d$ by (33), $\varphi(\hat{d}) > \gamma_{\hat{d}} = O(1)$ with probability one asymptotically. That is to say that the probability of miss detection is zero as $N \to \infty$. Knowing the limiting distribution of $\varphi(\hat{d})$ when $\hat{d} = d$, we can easily see that asymptotically the probability of false alarm (i.e., $\varphi(d) \geq \gamma_d$) is $1-\alpha$ (e.g., 1% or 0.1%), which can be set small enough. As shown in [22], the FSD detection scheme can be made *strongly consistent* (i.e., probability of error detection is strictly zero as $N \to \infty$), if the threshold $\gamma_{\hat{d}}$ is replaced by $\gamma_{\hat{d}} c(N)$, where $c(N)$ is a deterministic sequence that satisfies $c(N)/N \overset{N\to\infty}{\longrightarrow} 0$ and $c(N)/\log N \log N \overset{N\to\infty}{\longrightarrow} \infty$.

With $\varphi(\hat{d})$, it seems that we can also apply the *strongly consistent* MDL detection scheme at each Bi-Lanczos step. If we look more closely into the MDL scheme, there are two major difficulties. First, the MDL estimator is the *global* minimizers of its criterion:

$$MDL(\hat{d}) = L(\hat{d}) + \frac{1}{2}\hat{d}(2M-\hat{d}) \log N \quad (32)$$

for all possible $\hat{d} = 0, \cdots, M-1$. But at each Bi-Lanczos step, say the jth step, we can only obtain $\theta_i^{(j)}$, $i = 1, \cdots, j$ and therefore, we can at most evaluate $MDL(\hat{d})$ for $\hat{d} = 0, \cdots, j$. In other words, we do not seem to have sufficient information to apply the MDL procedure. Second, $\varphi(\hat{d})$ is *not* exactly the same as $L(\hat{d})$.

However, a new modified version of MDL, termed FSD–MDL [16], can be implemented at each intermediate Bi-Lanczos step. Suppose the new MDL criterion is $MDL_{FSD}(\hat{d}) = \varphi(\hat{d}) + \frac{1}{2}\hat{d}(2M - \hat{d})\log N$. Then the FSD–MDL detection scheme is as follows.

The FSD–MDL Detection Scheme

1) Set $\hat{d} = 0$.
2) Steps 2)-3) of the FSD detection scheme.
3) If $\varphi(\hat{d}) > \gamma_{\hat{d}}$, reject H$_0$; if $\hat{d} < m - 2$, $\hat{d} := \hat{d} + 1$, return to 2). Otherwise, continue with the Bi-Lanczos algorithm.
4) If $\varphi(\hat{d}) \leq \gamma_{\hat{d}}$, evaluate the partial MDL sequence, i.e., $MDL_{FSD}(k) = \varphi_k - k(M - k/2)\log(N)$, $k = 0, 1, \cdots, \hat{d}$. The estimate of d is the k that minimizes $MDL_{FSD}(k)$.

In fact, the FSD–MDL scheme can be considered as a combination of the FSD and the partial MDL schemes. Though the FSD scheme is not strongly consistent, its combination with MDL is. We shall prove this as follows. As shown in the above, asymptotically, the FSD scheme never underestimate d. In other words, if H$_0$ is accepted, $\hat{d} \geq d$ and $j \geq \hat{d} + 2 \geq d + 2$. In this case, by (31), $\varphi(k) = O(N)$ for $k < d$ and $\varphi(k) = O(1)$ for $k = d$. As for $k \geq d + 1$, since we can only show that $\theta_k^{(j)} - \hat{\sigma} = O(N^{-1/2})$, it is easy to see that $\varphi(k) = O(N^{1/2})$. Since $MDL_{FSD}(k) = \varphi(k) + p(k)$, where $p(k) = k(M - k/2)\log(N) = O(\log N)$, thus asymptotically,

$$MDL_{FSD}(k) = \begin{cases} \varphi(k) = O(N), & k < d \\ p(k) = O(\log N), & k = d\,. \\ \varphi(k) = O(N^{1/2}), & k > d \end{cases} \tag{33}$$

Thus, as $N \to \infty$, the minimizer of MDL_{FSD} is clearly d with probability one, therefore the MDL–FSD scheme is *strongly consistent.*

Since the MDL–FSD procedure is eventually corrected by the MDL scheme, we are not worried about the higher probability of over estimation. Therefore, the confidence value α can be set a little larger to allow faster termination of the algorithm. It is interesting to note that unusual increase of $MDL_{FSD}(k)$ for $k > d$ to $O(N^{1/2})$, which makes the notch of $MDL_{FSD}(k)$ at $k = d$ deeper. In other words, the difference of convergence among two classes of singular values in the Lanczos procedure helps the detection of their separation point at $k = d$.

D. Selection of an Initial Vector

The performance, e.g., convergence and numerical stability, of the Lanczos algorithms greatly depends on the choice of their initial vectors $\mathbf{f}$. Suppose that the initial vector happens to be chosen to be orthogonal to the signal subspace, then we will never obtain the signal subspace via the Bi-Lanczos algorithm. (In real computation, round-off errors will make a different story.) Clearly, it requires the knowledge of the eigenpairs to obtain a good $\mathbf{f}$. Usually, this knowledge is not available before we carry out the Lanczos-based algorithm. Nevertheless, as we have assumed, the signal subspace slowly varies and so does the eigenstructure of the covariance matrix. Hence, the singular values and right singular vectors of $\mathbf{X}_N(t)$ can be used as estimates of the singular values and right singular vectors of $\mathbf{X}_N(t+1)$, respectively. With this knowledge, we can apply a recently developed analysis result [22], [17] to obtain a good initial vector.

In [22], Xu and Kailath developed more explicit forms of (24) and (25), i.e.,

$$\frac{\hat{\sigma}_k^2 - \theta^{(j)}{}_k^2}{\hat{\sigma}_k^2 - \hat{\sigma}^2} \sim \left\{ \frac{\sin^2\angle(\hat{\mathcal{V}}^d, \mathbf{f})}{\cos^2\angle(\hat{\mathbf{v}}_k, \mathbf{f})} \prod_{\substack{i=1\\ i\neq k}}^{d} \frac{(\hat{\sigma}_i^2 - \hat{\sigma}^2)^2}{(\hat{\sigma}_i^2 - \hat{\sigma}_k^2)^2} \right\} \cdot \left(\frac{\epsilon}{\hat{\sigma}_k^2 - \hat{\sigma}^2} \right)^{2(j-d)} \tag{34}$$

$$\sin^2\angle(\mathbf{y}_k^{(j)}, \hat{\mathbf{v}}_k) \sim \left\{ \frac{\sin^2\angle(\hat{\mathcal{V}}^d, \mathbf{f})}{\cos^2\angle(\hat{\mathbf{v}}_k, \mathbf{f})} \prod_{\substack{i=1\\ i\neq k}}^{d} \frac{(\hat{\sigma}_i^2 - \hat{\sigma}^2)^2}{(\hat{\sigma}_i^2 - \hat{\sigma}_k^2)^2} \right\} \cdot \left(\frac{\epsilon}{\hat{\sigma}_k^2 - \hat{\sigma}^2} \right)^{2(j-d)} \tag{35}$$

where $(\hat{\sigma}_k, \hat{\mathbf{v}}_k)$ is the kth singular value and right singular vector of $\mathbf{R}$ or $\mathbf{X}_N(t+1)$, $\hat{\mathcal{V}}^d = span\{\hat{\mathbf{v}}_1, \hat{\mathbf{v}}_2, \cdots, \hat{\mathbf{v}}_d\}$, $\epsilon = \hat{\sigma}_{d+1}^2 - \hat{\sigma}_M^2$, and $(\theta^{(j)}{}_k, \mathbf{y}_k^{(j)})$ is the kth RR pair at the jth Bi-Lanczos step. In (34)-(35), $\hat{\sigma}^2 = \frac{1}{2}(\hat{\sigma}_{d+1}^2 + \hat{\sigma}_M^2)$ and $\epsilon = \hat{\sigma}_{d+1}^2 - \hat{\sigma}_M^2$. Since $\epsilon = O(N^{-1/2})$, (24) and (25) easily follow from (34) and (35).

As mentioned earlier, the Lanczos-type algorithms may have numerical problems. According to Paige's analysis [25], the numerical problems are essentially due to the ill-balanced convergence of the RR approximation. By (34) and (35), the estimation error depends on the distribution of the eigenvalues, as well as the initial vector $\mathbf{f}$. Though the distribution of the singular values are fixed, the initial vector $\mathbf{f}$ can be chosen to make the convergence well-balanced. In other words, when the estimation errors of the d signal eigenvalues are small enough, we want to make sure that the errors are close to one another to avoid any potential loss of orthogonality in the Lanczos algorithms. Of course, selection of an appropriate $\mathbf{f}$ is not possible if there is no *a priori* knowledge about the eigenstructure. If an estimation of the eigenstructure is available, we can use it in (34) and (35) to find a good initial vector $\mathbf{f}$, which can not only improve the numerical property of the Lanczos-based algorithms, but also significantly speed up the convergence.

Since the first RR pair has the smallest convergence rate $\epsilon/(\hat{\sigma}_1^2 - \hat{\sigma}^2)$, it usually converges the fastest, i.e., its estimation error is most likely the smallest. Suppose that at the mth step (e.g., $m = d + 2$), this error becomes small enough to potentially cause loss of orthogonality. To avoid such a numerical problem, we hope that all the other $d-1$ errors can be close to this so that the procedure will terminate, i.e.,

$$\begin{aligned} &\frac{\hat{\sigma}_k^2 - \theta_k^{(m)}}{\hat{\sigma}_k^2 - \sigma} \approx \frac{\hat{\sigma}_1^2 - -\theta_1^{(m)}}{\hat{\sigma}_1^2 - \hat{\sigma}}, \\ &\sin \angle(\mathbf{y}_k^{(m)}, \hat{\mathbf{v}}_k) \approx \sin \angle(\mathbf{y}_1^{(m)}, \hat{\mathbf{v}}_1), \\ &2 \le k \le d. \end{aligned} \tag{36}$$

The basic idea is to construct an initial vector $\mathbf{f}$ such that the initial error of the fastest converging RR pair become the largest, and *vice versa*. By (34) and (35),

$$\begin{aligned} &\left\{ \frac{\sin^2 \angle(\hat{\mathcal{V}}^d, \mathbf{f})}{\cos^2 \angle(\hat{\mathbf{v}}_k, \mathbf{f})} \prod_{\substack{i=1 \\ i \neq k}}^{d} \frac{(\hat{\sigma}_i^2 - \hat{\sigma}^2)^2}{(\hat{\sigma}_i^2 - \hat{\sigma}_k^2)^2} \right\} \cdot \left(\frac{\epsilon}{\hat{\sigma}_k^2 - \hat{\sigma}^2} \right)^{2(m-d)} \\ &= \left\{ \frac{\sin^2 \angle(\hat{\mathcal{V}}^d, \mathbf{f})}{\cos^2 \angle(\hat{\mathbf{v}}_1, \mathbf{f})} \prod_{i=2}^{d} \frac{(\hat{\sigma}_i^2 - \hat{\sigma}^2)^2}{(\hat{\sigma}_i^2 - \hat{\sigma}_1^2)^2} \right\} \cdot \left(\frac{\epsilon}{\hat{\sigma}_1^2 - \hat{\sigma}^2} \right)^{2(m-d)}. \end{aligned}$$

Some elementary calculation leads to

$$\cos \angle(\hat{\mathbf{v}}_k, \mathbf{f}) = c_k / c_1 \cdot \cos(\hat{\mathbf{v}}_1, \mathbf{f}), \quad 2 \le k \le d \tag{37}$$

where

$$c_k = \left(\prod_{\substack{i=1 \\ i \neq k}}^{d} |\hat{\sigma}_i^2 - \hat{\sigma}_k^2| \cdot (\hat{\sigma}_k^2 - \hat{\sigma}^2)^{m-d+1} \right)^{-1}, \quad 1 \le k \le d. \tag{38}$$

It is also desired to make all the errors small or $\sin \angle(\hat{\mathcal{V}}^d, \mathbf{f}) \approx 0$. Thus,

$$\sum_{k=1}^{d} \cos^2 \angle(\hat{\mathbf{v}}_k, \mathbf{f}) = 1. \tag{39}$$

By (37)–(39), it is easy to obtain

$$\cos \angle(\hat{\mathbf{v}}_k, \mathbf{f}) = \frac{c_k}{\sqrt{c_1^2 + c_2^2 + \cdots + c_d^2}} \tag{40}$$

and

$$\mathbf{f} = \sum_{k=1}^{d} \hat{\mathbf{v}}_k \cdot \cos \angle(\hat{\mathbf{v}}_k, \mathbf{f}). \tag{41}$$

Obviously, $(\hat{\sigma}_k, \hat{\mathbf{v}}_k)_{k=1}^d$ of $\mathbf{X}_N(t+1)$ are not available in real applications. However, since they vary slowly in time, the principal RR pairs $(\theta_k^{(j)}, \mathbf{y}_k^{(j)})_{k=1}^d$ of $\mathbf{X}_N(t)$ can serve as their rough estimates. $\hat{\sigma}$ can be estimated by either $\left(\prod_{k=1}^{d} \theta_k^{(j)^2}\right)^{1/(M-d)}$ or $\frac{1}{M-d}\left(\|\mathbf{R}\|_F^2 - \sum_{k=1}^{d} \theta_k^{(j)^2}\right)$. Substituting these estimates of $(\hat{\sigma}_k, \hat{\mathbf{v}}_k)_{k=1}^d$ into (38) and (41), we can obtain a good initial vector $\mathbf{f}$.

From (41), it is easy to see that $\mathbf{f}$ lies approximately in the signal subspace. Such an $\mathbf{f}$ may lead to some worry that it can restrict the search of the signal subspace and thus cannot detect the increase of the signal subspace dimension. This is, in fact, not true in general. If there is a new signal coming in with new array manifold, then this vector can alter the whole eigenvectors and with probability one, the new eigenvector is not necessarily orthogonal to $\mathbf{f}$. As long as $\mathbf{f}$ is not orthogonal to any principal eigenvector, it can be used to update the signal subspace estimate, even if the signal subspace dimension increases. This claim is verified by the simulation results in Section V.

E. Robust Bidiagonalization Based on Jacobi Rotations

Although the numerical properties of the Bi-Lanczos algorithm can be significantly improved by choosing a good initial vector, numerical stability is still not guaranteed. For some applications that require absolute numerical stability, we propose an alternative to the Bi-Lanczos method using Jacobi rotations.

Since the goal of the Bi-Lanczos algorithm is to partially bidiagonalize a matrix, there are certainly other means of achieving this. One numerically stable approach [17] is to use Householder transformations and Jacobi rotations. The basic idea is described below. First, we carry out a row Householder reflection based on the first column of $\mathbf{R}$ or $M-1$ row Jacobi rotations such that the first column is made zero except for the first element. Then we apply the orthogonal column transformations on the resultant matrix starting from the second column such that all the elements of the first row but the first two become zero. Clearly, since we do not touch the first column, its zero elements remain the same. So far, we have completed the first step of the bidiagonalization. Continuing the above procedure, we can eventually bidiagonalize the whole matrix: $\mathbf{R} = \mathbf{U}\mathbf{B}\mathbf{V}^H$. The whole procedure is illustrated in (42), where the Householder

$$\begin{aligned} &\begin{bmatrix} \times & \times & \cdots & \times \\ \times & & & \\ \times & & & \\ \vdots & & \mathbf{R}_0 & \\ \times & & & \end{bmatrix} \xrightarrow[\mathbf{H}_1^r]{\text{RowOp.}} \begin{bmatrix} \alpha_1 & \times & \cdots & \times \\ 0 & & & \\ 0 & & & \\ \vdots & & \mathbf{R}_1 & \\ 0 & & & \end{bmatrix} \xrightarrow[\mathbf{H}_2^c]{\text{Col.Op.}} \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ 0 & \times & \times & \cdots & \times \\ 0 & \times & & & \\ \vdots & \vdots & & \mathbf{R}_1' & \\ 0 & \times & & & \end{bmatrix} \\ &\xrightarrow[\mathbf{H}_2^r]{\text{RowOp.}} \begin{bmatrix} \alpha_1 & \beta_1 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \times & \cdots & \times \\ 0 & 0 & & & \\ 0 & 0 & & & \\ \vdots & \vdots & & \mathbf{R}_2 & \\ 0 & 0 & & & \end{bmatrix} \xrightarrow[\mathbf{H}_3^c]{\text{Col.Op.}} \begin{bmatrix} \alpha_1 & \beta_1 & 0 & 0 & \cdots & 0 \\ 0 & \alpha_2 & \beta_2 & 0 & \cdots & 0 \\ 0 & 0 & \times & \times & \cdots & \times \\ 0 & 0 & \times & & & \\ \vdots & \vdots & \vdots & & \mathbf{R}_2' & \\ 0 & 0 & \times & & & \end{bmatrix} \cdots \end{aligned} \tag{42}$$

transformation is applied. (See (42) at the bottom of the previous page.)

Unlike the Bi-Lanczos algorithm, which may be numerically unstable, such an approach relies on orthogonal transformations and is always numerically stable. However, the Lanczos algorithm can start with any initial vector $\mathbf{f}$, which is also the first column of $\mathbf{V}$, i.e, $\mathbf{f} = \mathbf{V}\mathbf{e}_1$. Since the row transformation begins from the second column, it is not too difficult to see that $\mathbf{V}\mathbf{e}_1 = \mathbf{e}_1$ [17]. It seems that the rotation procedure always starts with $\mathbf{f} = \mathbf{e}_1$. Now the question is whether we can make this rotation approach start with an arbitrary initial vector $\mathbf{f}$. In Section IV-D, we have shown how to compute a good $\mathbf{f}$ such that the convergence of the RR approximation is fast and well-balanced. In the following, we shall show how to start the rotation procedure with this good initial vector by one more bidiagonalization step.

First, let $\mathbf{G}$ denote either the Householder reflection or a combination of $M-1$ Jacobi rotations that transform $\mathbf{f}$ to $\mathbf{e}_1$, i.e., $\mathbf{e}_1 = \mathbf{G}^H\mathbf{f}$ or $\mathbf{f} = \mathbf{G}\mathbf{e}_1$. Now, let us apply the same $\mathbf{G}$ on $\mathbf{R}$, i.e., $\mathbf{R}' = \mathbf{R}\mathbf{G}$ and then implement the rotation approach (via Householder reflections or Jacobi rotations) to bidiagonalize $\mathbf{R}'$: $\mathbf{R}' = \mathbf{U}'\mathbf{B}'\mathbf{V}'^H$. Then $\mathbf{R} = \mathbf{U}'\mathbf{B}'\mathbf{V}'^H\mathbf{G}^H$. Clearly the $\mathbf{V}$ for $\mathbf{R}$ is $\mathbf{G}\mathbf{V}'$ and

$$\mathbf{V}\mathbf{e}_1 = \mathbf{G}\mathbf{V}'\mathbf{e}_1 = \mathbf{G}\mathbf{e}_1 = \mathbf{f}. \tag{43}$$

Therefore, by applying an additional rotation $\mathbf{P}$, we can make the rotation procedure start with an arbitrary initial vector $\mathbf{f}$.

So far, we have not exploited the upper-triangular structure of $\mathbf{R}$. It turns out that such a structure can be exploited to reduce the computational complexity. The crucial point is to maintain the upper-triangular structure at each bidiagonalization step. Let us start with nulling out the last element of the first row by applying a Jacobi rotation to the last two columns. Then, the upper-triangular structure is immediately lost since the $(M, M-1)$ element is non-zero. In this case, we can apply a row rotation on the last two rows to zero out this element and regain the upper-triangular structure. Then, the $(1, M-1)$ element is nulled out by the rotating the $(M-2)$th and $(M-1)$th columns. We can apply another row rotation to regain the matrix upper-triangular again ... The flowchart at the bottom of the page shows how the first row of $\mathbf{R}$ is bidiagonalized by switching column and row rotations. Since $\mathbf{R}$ is upper-triangular, in the above transformations, the size of the column transformations vary from M to 2 while that of the row transformations vary from 2 to M. Therefore, the total flop count for determining the partial bidiagonal form (size j) is about $2\frac{1}{2}Mj = Mj$ Jacobi rotations with size M, while the cost is about $2Mj$ Jacobi rotation if the upper-triangular structure is not exploited. However, the computation required for accumulating the $\mathbf{V}$ remains the same. Clearly, since exploiting the upper-triangular structure requires more selective nulling, Householder reflections are not applicable and only Jacobi rotations can be used.

Now, let us show how we handle an arbitrary initial vector $\mathbf{f}$. As mentioned before, we need to first find $\mathbf{G}$ that transforms $\mathbf{f}$ to $\mathbf{e}_1$. Clearly, $\mathbf{G}$ can be a product of $M-1$ Jacobi rotations starting from the last element of $\mathbf{f}$. Applying $\mathbf{G}$ to $\mathbf{R}$ on the right will definitely destroy its upper-triangular form. Nevertheless, following the idea of the above approach, we can also carry out appropriate row transformations to regain the upper-triangular structure.

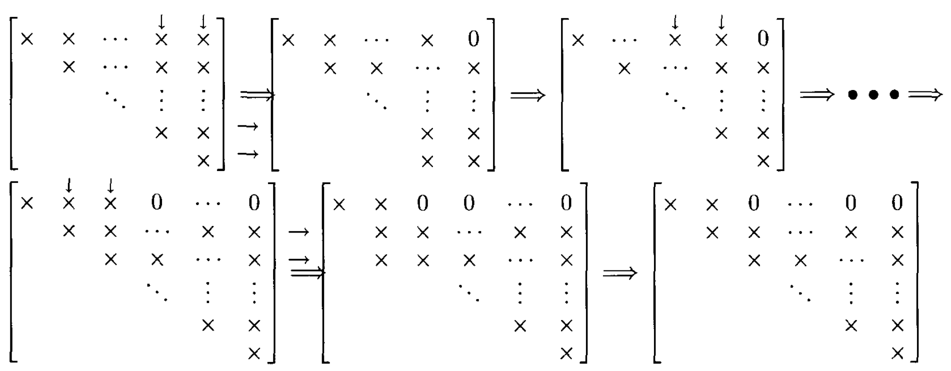
Although the rotation methods are numerically stable, their computational cost is a little higher than the standard Bi-Lanczos algorithm. Suppose all the methods terminate at the $j = O(d)$ steps, then Bi-Lanczos requires $2M^2j$ flops since the most significant computation at each step is two matrix-vector products. Exploiting the fact that nearly half of the elements of $\mathbf{R}$ are zero, the computational complexity of Bi-Lanczos is about M^2j. As we know, the cost of a size-M Householder reflection and (fast) Jacobi rotation (including accumulating $\mathbf{V}$) takes about $4M^2$ and $6M$ flops[3] respectively. Therefore, the procedure using j Householder transformations or Mj Jacobi rotations require $4M^2j$ and $6M^2j$ flops. Overall, all the above mentioned methods are $O(M^2d)$ algorithms; the methods relying on rotations have larger coefficients of complexity but they are numerically superior.

We are now in a position to summarize the major steps of the Lanczos based algorithms.

The Bi-Lanczos Based Algorithm:

1) For every n new data vectors, update and downdate the Cholesky factor $\mathbf{R}$ of the sample covariance matrix (see Section III for details).

[3]It usually requires $12M^2$ flops to achieve a j partial bidiagonalization (including accumulating $\mathbf{V}$). Here, $\mathbf{R}$ is an upper-triangular matrix which is exploited to reduce the computational complexity by half.



2) Find the good starting vector **f** by (41) based on the previous signal eigenpairs.
3) Apply the Bi-Lanczos algorithm or its more stable alternatives at each Lanczos step, employ Bi-FSD or FSD–MDL detection schemes to estimate the signal subspace dimension d (see Section IV-D for details).

The Tri-Lanczos Based Algorithm:

1) For every n new data vectors, directly update and downdate the sample covariance matrix $\mathbf{C_x}(t)$ by (11).
2) Find the good starting vector **f** by (41) based on the previous signal eigenpairs.
3) Apply the Tri-Lanczos algorithm or its more stable alternatives and at each Lanczos step, employ Tri-FSD detection schemes to estimate the signal subspace dimension d (see Section IV-D for details).

V. Numerical Simulations

In this section, we present some numerical simulation results to demonstrate the performance of the new Lanczos-based update algorithms. In all the three scenarios we simulated, a number of narrow-band plane waves impinges on a 20-element uniform linear array with a half wavelength separation. The observed data also contains additive and spatially white noise. Initially, at $t = 0$, there are three sources with directions-of-arrival (DOA's) 10°, 30°, and 75° and with signal-to-noise-ratios (SNR's) −2, 0, and −5 dB, respectively. The SNR in this paper is defined as the ratio of the amplitude of each individual source versus the standard deviation of the noise. We assume that the amplitudes of signal and noise remain the same while the DOA's vary gradually as shown in Fig. 3. In the first scenario, three sources were present all the time, while in another scenario some sources were turned off and on. The covariance matrix and its Cholesky factor were updated for each new snapshot. The signal subspace estimation was carried out once every five snapshots due to the slow variation of the signal subspace. The Tri-Lanczos method (or Tri-Lanczos) method was applied to the covariance matrix, whereas the Bi-Lanczos method (or Bi-Lanczos) was used to estimate the signal subspace from the Cholesky factor of the covariance matrix. Both Tri-Lanczos and Bi-Lanczos start with the pre-calculated initial vector **f**. Tri-Lanczos is terminated if the Tri-FSD detection scheme [22], [17] is completed, while the Bi-Lanczos based method ends when the Bi-FSD detection scheme is done. However, the estimate of d is the minimizer of the partial MDL (or FSD–MDL) scheme. The SVD was also computed for comparison. The quality of the signal subspace estimates is evaluated by the error of the DOA estimates, which were estimated using the ESPRIT algorithm [26]–[27].

In the first example, we used many different detection schemes to first estimate the signal subspace dimension. In the Tri-Lanczos based method, we applied the Tri-FSD detection scheme based on the ratio of the quadratic and arithmetic means of the smaller eigenvalues (or squares of the corresponding singular values) [16], [17], [22]. In the Bi-Lanczos based method, we tried the FSD likelihood ratio (LR) test and also the FSD–MDL scheme, which are described in Section

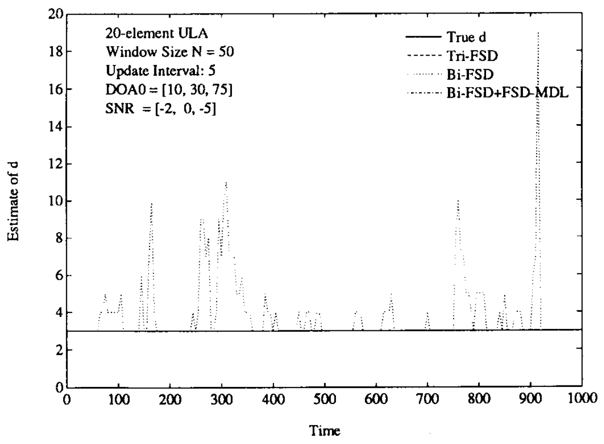


Fig. 1. Estimation of d using Tri-FSD, Bi-FSD, Bi-Lanczos+FSD–MDL (example 1).

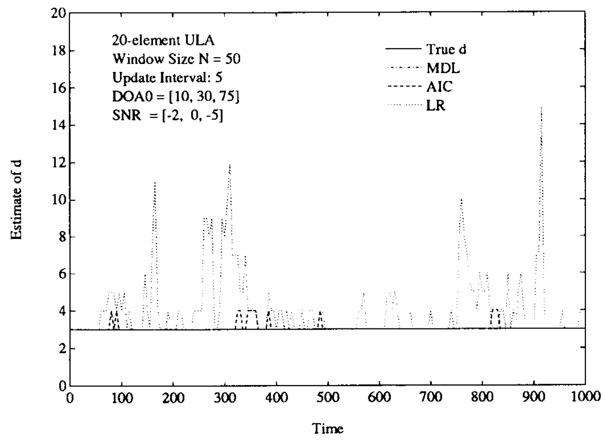


Fig. 2. Estimation of d using MDL, AIC, and LR (example 1).

IV-D. After we obtain all the eigenvalues via the SVD of **R**, we used the MDL, AIC and LR detection schemes and the results are shown in Fig. 2. From these figures, we see that the likelihood ratio tests (LR and Bi-FSD) based on the ratio of geometric and arithmetic means are in general not accurate. As analyzed in Section IV-D, these tests tend to overestimate d. On the other hand, MDL and FSD–MDL (implemented with Bi-Lanczos) performed quite well, and it always gives a correct d estimate (refer to Figs. 1 and 2). It is also interesting to note that the Tri-FSD scheme also did not make any mistake in detecting d. The AIC detection scheme behave reasonably well; but it makes mistake occasionally.

Fig. 3 shows the true DOA's and the DOA estimates obtained from the Tri-Lanczos method applied to the covariance matrix $\mathbf{R}_x$. The results of Bi-Lanczos and SVD applied to **R** are shown in Figs. 4 and 5. From these plots, it is easily seen that the Lanczos-based algorithms (Tri-Lanczos and Bi-Lanczos) work almost equally well as a full SVD. The computational complexity of these algorithms is listed in Table

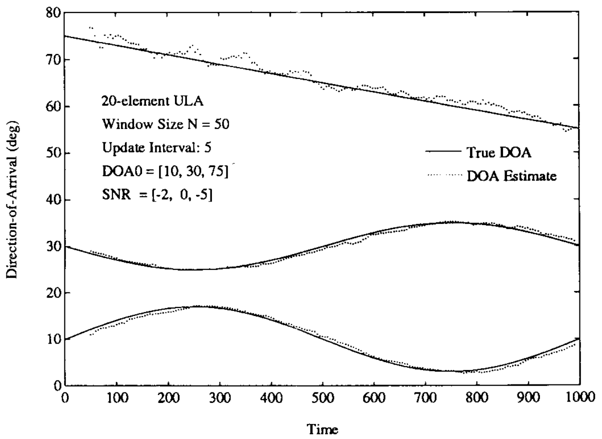


Fig. 3. True DOA's versus the DOA estimates from Tri-Lanczos (example 1).

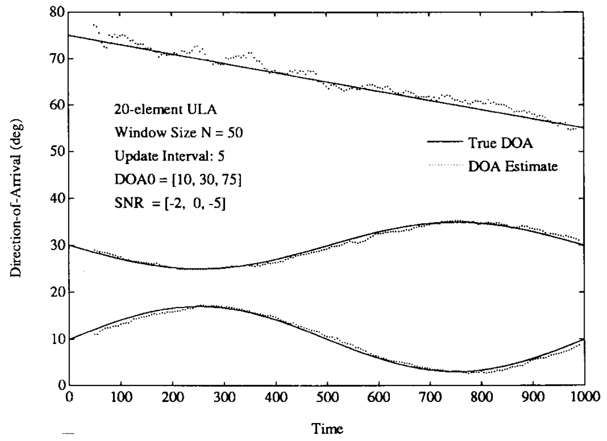


Fig. 4. True DOA's versus the DOA estimates from Bi-Lanczos (example 1).

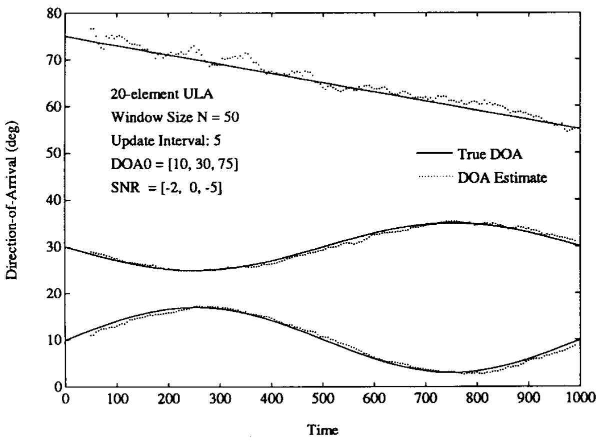


Fig. 5. True DOA's versus the DOA estimates from SVD (example 1).

TABLE I
EXPERIMENT PARAMETERS IN THE SIMULATIONS

# of Elements (M)	20 (with $\frac{1}{2}\lambda$ separation)
# of Sources (d)	3 (with time-varying DOA's)
Starting DOA's	$10°$, $30°$, $75°$
Window Size (N)	50 data vectors
SNR's	-2dB, 0dB, -5dB
Update Interval (n)	5 data vectors

TABLE II
COMPARISON OF COMPUTATIONAL COMPLEXITY OF THREE METHODS: BI-LANCZOS, TRI-LANCZOS, AND SVD (a) EXAMPLE 1, (b) EXAMPLE 2

Computation Complexity	Tri-Lanczos	Bi-Lanczos	SVD
Average number of steps	5	5.88	-
Average flops	13600	27318	197461
Comp. saving over SVD	15	7	1

(a)

Computation Complexity	Tri-Lanczos	Bi-Lanczos	SVD
Average number of steps	4.53	5.44	-
Average flops	12584	24857	199128
Comp. saving over SVD	16	8	1

(b)

II, from which we can see that the computational savings of the Lanczos approach is quite significant (a factor of 15 and 7). On average, Bi-Lanczos takes about one more step than Tri-Lanczos. The reason is that the Bi-Lanczos algorithm relies on the Bi-FSD detection rule followed by the FSD–MDL scheme. Since the Bi-FSD criterion tends to over-estimate d, the termination of the algorithm can be delayed. We also note that the computational cost of Bi-Lanczos is twice as much as that of Tri-Lanczos. This is because Tri-Lanczos uses the traditional Lanczos algorithm while Bi-Lanczos relies on the numerically more stable Jacobi rotations. If the Bi-Lanczos algorithm is applied to $\mathbf{R}$, its computational complexity can be similar to that of Tri-Lanczos. The simulation results (refer Table II(a)) also show that the Lanczos-based algorithms converge in roughly $5 = (d + 2)$ Lanczos steps on average, which is consistent with our theoretical results. In reality, since we only have 50 data samples and the scenario is far from the asymptotic region, conventional Lanczos algorithms without the good initial vector usually take a few more steps (e.g., 6 or 7 steps) to achieve convergence. The reason that the proposed algorithms converge so fast should be attributed to the choice of a good initial vector $\mathbf{f}$. In summary, the results of this example indicate that the fast Bi-Lanczos and Tri-Lanczos algorithms achieve equivalent performance to the more costly SVD based approach.

The second simulation scenario is more difficult as some sources were turned off and on (refer to Fig. 8), which is often the case in mobile communications. This scenario can be used to test the statistical robustness of our tracking algorithms. Without a detection scheme, tracking algorithms may not be

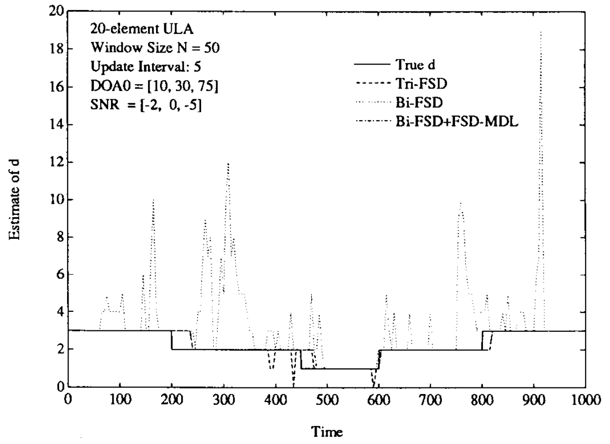


Fig. 6. Estimation of d using Tri-FSD, Bi-FSD, Bi-FSD+FSD–MDL (example 2).

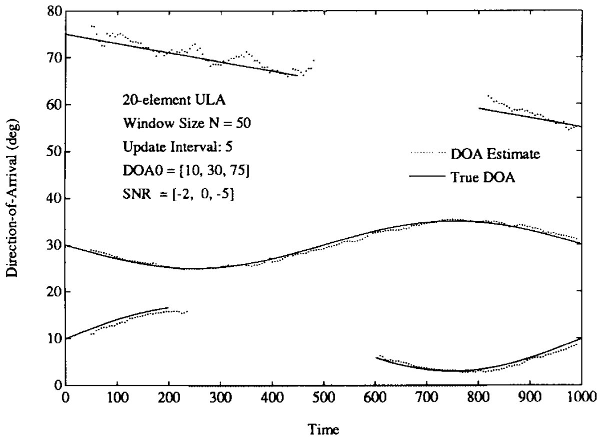


Fig. 8. True DOA's versus the DOA estimates from Tri-FSD (example 1).

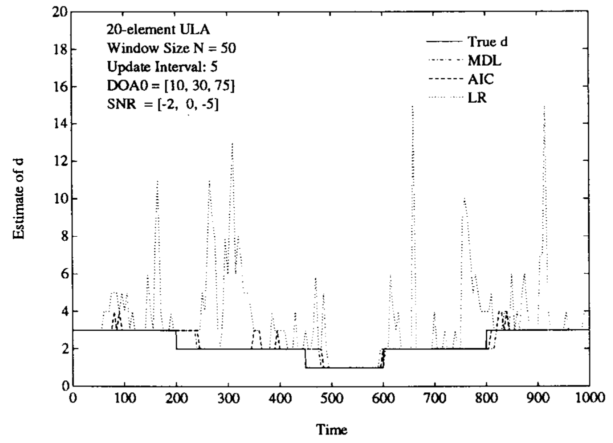


Fig. 7. Estimation of d using MDL, AIC, and LR (example 2).

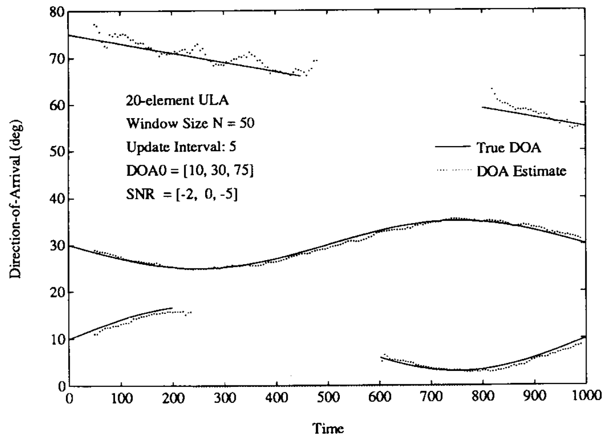


Fig. 9. True DOA's versus the DOA estimates from Bi-FSD (example 1).

so useful to track the signal subspace. As mentioned earlier, most existing fast tracking algorithms [4] make no mention about robust methods for detecting the dimension of the signal subspace. With the proposed detection schemes developed in the above sections, we can also update the signal subspace dimension in the process of updating the signal subspace. The results of the detection schemes are shown in Figs. 6 and 7. These detection schemes perform similarly as in the first scenario. The best schemes are MDL, Bi-FSD + FSD–MDL and Tri-FSD. Apart from some delay effects, the schemes track d quite successfully and they all perform equally well except that Tri-FSD scheme underestimated[4] d a few times (refer to Fig. 6). The DOA estimation and computational complexity (refer to Figs. 8–10 and Table II(b)) are quite similar to those in the first example. Overall, these simulation results indicate that the proposed fast algorithms can adapt to abrupt changes of the signal subspace size and can track the signal subspace in a robust way.

[4]This may be due to the numerical problems of the Tri-Lanczos algorithm.

VI. Concluding Remarks

In this paper, we presented a class of fast $O(M^2d)$ algorithms (FSD) for tracking the low-dimensional signal subspace. These algorithms are based on partial tridiagonalization or bidiagonalization procedures including the well-known Lanczos algorithms and they achieve almost an order of magnitude of computational complexity reduction over the $O(M^3)$ SVD based methods. Several detection schemes were also presented that can be implemented at each intermediate step of bidiagonalization or tridiagonalization. Certain implementation details have been discussed such as a way of exploiting the knowledge of an approximation of the signal subspace to find a good initial vector and more numerically stable algorithms for bidiagonalization and tridiagonalization of the data or covariance matrices. Under certain stationarity assumptions, we also showed that the performance of our fast tracking algorithms is *asymptotically equivalent* to that of the more costly SVD and the detection schemes are *strongly consistent.* Computer simulation has been conducted and the results verified our theoretical claims about the FSD approach. Since the

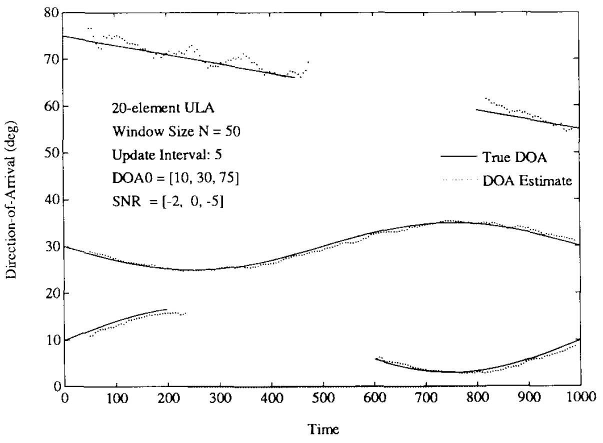


Fig. 10. True DOA's versus the DOA estimates from SVD (example 2).

most computationally intensive part of the Lanczos algorithms is $O(d)$ matrix-vector products, the proposed algorithms are amenable to parallelization: with $O(M)$ processors, we can easily reduce their computation time from $O(M^2d)$ to $O(Md)$, which looks very attractive for real-time implementations.

References

[1] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, pp. 31–48, 1978.

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

[3] G. H. Golub and C. F. Van Loan, *Matrix Computations.* Baltimore, MD: Johns Hopkins Univ. Press, 1990.

[4] P. Comon and G. Golub, "Tracking a few extreme singular values and vectors in signal processing," *Proc. IEEE*, vol. 78, pp. 1327–1343, Aug. 1990.

[5] G. Golub and W. Kahan, "Calculating the singular values and pseudo-inverse of a matrix,"*SIAM J. Numerical Anal.*, vol. 2, pp. 205–224, 1965.

[6] P. Comon, "Fast updating of a low-rank approximation to a varying hermitian matrix," in *Proc. 22nd Asilomar Conf. Sig., Syst. Comput.*, Pacific Grove, CA, Nov. 1988, vol. 1, pp 358–362.

[7] J. F. Yang and M. Kaveh, "Adaptive eigensubspace algorithms for direction or frequency estimation and tracking," *IEEE Trans. Acoust., Speech, Signal Processing* , vol. ASSP-36, pp. 241–251, Feb. 1988.

[8] D. R. Fuhrmann, "An algorithm for subspace computation with applications in signal processing," *SIAM J. Matrix Anal. App.*, vol. 9, no. 1, 1988.

[9] W. R. Ferng, G. H Golub, and R. J. Plemmons, "Lanczos methods for recursive condition estimation," *Numerical Algorithm*, vol. 1, pp. 1–20, 1991.

[10] G. W. Stewart, "An updating algorithm for signal subspace tracking," Tech. Rep. CS-TR 2494, Univ. Maryland, Comput. Sci. Tech. Rep., College Park, MD, 1990.

[11] M. F. Griffin and G. W. Stewart, "Updating MUSIC and root-MUSIC with the rank-revealing URV decomposition," in *Proc. 25th Asilomar Conference on Sig., Syst. Comput.*, Pacific Grove, CA, Nov. 19 91, pp. 277–281.

[12] M. F. Griffin, E. C. Boman, and G. W. Stewart, "Minimum-norm updating with the rank-revealing URV decomposition," in *Proc. ICASSP'92*, San Francisco, CA, Apr. 1992, pp. 293–296.

[13] I. Karasalo, "Estimating the covariance matrix by signal subspace averaging," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 8–12, Jan. 1986.

[14] R. DeGroat and R. Roberts, "Efficient, numerically stabilized rank-one eigenstructure updating," *IEEE Trans. Acoust., Speech, Signal Processing* , vol. ASSP-34, pp. 301–316, Oct. 1986.

[15] R. DeGroat, "Noniterative subspace tracking," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, pp. 571–577, Mar. 1992.

[16] G. Xu and T. Kailath, "A fast algorithm for signal subspace decomposition and its performance analysis," in *Proc. ICASSP 91 Conf.*, Toronto, ON, Canada, May 1991, vol. 5, pp. 3069–3072.

[17] G. Xu, "Fast subspace decomposition and its applications," Ph. D. dissertation, Stanford University, Stanford, CA, Sept. 1991.

[18] T. W. Anderson, "Asymptotic theory for principal component analysis," *Ann. Math. Statist.*, vol. 34, pp. 122–148, 1963.

[19] G. H. Golub, "Matrix decompositions and statistical computation," in *Statistical Computation*, R. C. Milton and J. A. Nelder, eds. New York: Academic Press, 1969, pp. 365–397.

[20] M. Wax and T. Kailath, "Detection of signals by information theoretic criteria," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 387–392, Apr. 1985.

[21] W. F. Stout, *Almost Sure Convergence.* New York: Academic Press, 1974.

[22] G. Xu and T. Kailath, "Fast subspace decomposition," *IEEE Trans. Sig. Proc.*, vol. 42, pp. 539–551, Mar. 1994.

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

[24] R. Onn, A. Steinhardt, and A. Bojanczyk, "Hyperbolic singular value decomposition and applications," *IEEE Trans. Sig. Proc.*, vol. 39, July 1991.

[25] C. C. Paige, "The computation of eigenvalues and eigenvectors of very large sparse matrices," Ph. D. dissertation, Univ. London, London, U.K., 1971.

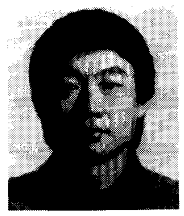
[26] A. Paulraj, R. Roy, and T. Kailath, "A subspace rotation approach to signal parameter estimation}," *Proc. IEEE*, vol. 74, pp. 1044–1045, July 1986.

[27] R. Roy and T. Kailath, "ESPRIT—Estimation of signal parameters via rotational invariance techniques," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 32, pp. 984–995, July 198 9.

Guanghan Xu (S'86–M'92) received the B. S. degree (Hons) in biomedical engineering from Shanghai Jiao Tong University, Shanghai, China in 1985; the M. S. degree in electrical engineering from Arizona State University, Tempe, AZ, in 1988; and the Ph. D. degree in electrical engineering from Stanford University, Stanford, CA, in 1991.

During the summer of 1989, he was a Research Fellow at the Institute of Robotics, the Swiss Institute of Technology, Zurich, Switzerland. From 1990 to 1991, he was a General Electric Fellow of the Fellow-Mentor-Advisor Program at the Center of Integrated systems, Stanford University. From 1991 to 1992, he was a Research Associate in the Department of Electrical Engineering, Stanford University, and a short-term visiting scientist at the Laboratory of Information and Decision systems of MIT. In 1992, he joined the faculty of the Department of Electrical and Computer Engineering at the University of Texas at Austin. He has worked in several areas including signal processing, communications, numerical linear algebra, multivariate statistics, and semiconductor manufacturing. His current research interest is applying advanced signal processing techniques to wireless communications and semiconductor manufacturing.

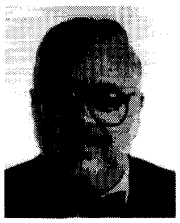
Dr. Xu is a member of Phi Kappa Phi.



Hongyuan Zha (PM'93) received the B. S. degree in mathematics from Fudan University, Shanghai, China in 1984 and the Ph. D. degree in scientific computing from Stanford University in 1993.

From 1988 to 1989, he was a visiting research assistant at Konrad-Zuse Zentrum für Informationstechnik, Berlin, Germany and ESAT Laboratory, Dept. of Electrical Engineering, K. U. Leuven, Belgium. In 1992, he joined the Department of Computer Science and Engineering at Pennsylvania State University. His research interests are numerical linear algebra and applications in statistics and signal processing.

Dr. Zha is a member of SIAM.



Gene H. Golub (HM'93) attended the University of Illinois where he received the 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 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 (1985–1987). 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.

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

Thomas Kailath (S'57–M'62–F'70) received the S. M. degree in Poona, India, and the Sc. D degree at the Massachusetts Institute of Technology in 1959 and 1961, respectively.

From October 1961 to December 1962, he worked at the Jet Propulsion Laboratories, Pasadena, CA, where he also taught part time at the California Institute of Technology. He then came to Stanford University where he served as Director of the Information Systems Laboratory from 1971–1980 and then as Associate Department Chairman from 1981 on. He currently holds the Hitachi America Professorship in Engineering. He has held short-term appointments at several institutions around the world. He has worked in a number of areas including information theory, communications, computation, control, signal processing, VLSI design, statistics, linear algebra and operator theory. His recent interests include applications of signal processing, computation, and control to problems in semiconductor manufacturing and wireless communication. He is the author of *Linear Systems* (Prentice Hall, 1980); and *Lectures on Wiener and Kalman Filtering*, (Springer-Verlag, 1981).

Dr. Kailath has held Guggenheim, Churchill, and Royal Society Fellowships, among others, and received awards from the IEEE Information Theory Society and the American Control Council, in addition to the Technical Achievement and Society Awards of the IEEE Signal Processing Society. He served as President of the IEEE Information Theory Society in 1975, and has been awarded honorary doctorates by Linköping University, Sweden, and by Strathclyde University, Scotland. He is a Fellow of the IEEE and of the Institute of Mathematical Statistics and is a member of the National Academy of Engineering..