Hindawi Publishing Corporation
EURASIP Journal on Advances in Signal Processing
Volume 2009, Article ID 576972, 16 pages
doi:10.1155/2009/576972

Research Article

Fast Subspace Tracking Algorithm Based on the Constrained Projection Approximation

Amir Valizadeh[1, 2] and Mahmood Karimi (EURASIP Member)[1]

[1] *Electrical Engineering Department, Shiraz University, 713485 1151 Shiraz, Iran*
[2] *Engineering Research Center, 134457 5411 Tehran, Iran*

Correspondence should be addressed to Amir Valizadeh, amirvalizadeh81@yahoo.com

Received 19 May 2008; Revised 4 November 2008; Accepted 28 January 2009

Recommended by J. C. M. Bermudez

We present a new algorithm for tracking the signal subspace recursively. It is based on an interpretation of the signal subspace as the solution of a constrained minimization task. This algorithm, referred to as the constrained projection approximation subspace tracking (CPAST) algorithm, guarantees the orthonormality of the estimated signal subspace basis at each iteration. Thus, the proposed algorithm avoids orthonormalization process after each update for postprocessing algorithms which need an orthonormal basis for the signal subspace. To reduce the computational complexity, the fast CPAST algorithm is introduced which has $O(nr)$ complexity. In addition, for tracking the signal sources with abrupt change in their parameters, an alternative implementation of the algorithm with truncated window is proposed. Furthermore, a signal subspace rank estimator is employed to track the number of sources. Various simulation results show good performance of the proposed algorithms.

1. Introduction

Subspace-based signal analysis methods play a major role in contemporary signal processing area. Subspace-based high-resolution methods have been developed in numerous signal processing domains such as the MUSIC, the minimum-norm, the ESPRIT, and the weighted subspace fitting (WSF) methods for estimating frequencies of sinusoids or directions of arrival (DOA) of plane waves impinging on a sensor array. In wireless communication systems, subspace methods have been employed for channel estimation and multiuser detection in code division multiple access (CDMA) systems. The conventional methods for extracting the desired information about the signal and noise subspaces are achieved by either the eigenvalue decomposition (EVD) of the covariance data matrix or the singular value decomposition (SVD) of the data matrix. However, the main drawback of these conventional decompositions is their inherent complexity.

In order to overcome this difficulty, a large number of approaches have been introduced for fast subspace tracking in the context of adaptive signal processing. A well-known method is Karasalo's algorithm [1], which involves the full SVD of a small matrix. A fast tracking method (the FST algorithm) based on the Givens rotations is proposed in [2]. Most of other techniques can be grouped into several families. One of these families includes classical batch methods for EVD/SVD such as QR-iteration algorithm [3], Jacobi SVD algorithm [4], and power iteration algorithm [5], which have been modified to fit adaptive processing. Other matrix decompositions have also successfully been used in subspace tracking. The rank-revealing QR factorization [6], the rank-revealing URV decomposition [7], and the Lankzos-diagonalization [8] are some examples of this group. In another family, variations and extensions of Bunch's rank-one updating algorithm [9], such as subspace averaging [10], have been proposed. Another class of algorithms considers the EVD/SVD as a constrained or unconstrained optimization problem, for which the introduction of a projection approximation leads to fast subspace tracking methods such as PAST [11] and NIC [12] algorithms. In addition, several other algorithms for subspace tracking have been developed in recent years.

Some of the subspace tracking algorithms add orthonormalization step to achieve orthonormal eigenvectors [13], which increases the computational complexity. The necessity of orthonormalization depends on the post-processing method which uses the signal subspace estimate to extract the desired signal information. For example, if we are using MUSIC or minimum-norm method for estimating DOA's or frequencies from the signal subspace, the orthonormalization step is crucial, because these methods need an orthonormal basis for the signal subspace.

From the computational point of view, we may distinguish between methods having $O(n^3)$, $O(n^2r)$, $O(nr^2)$, or $O(nr)$ operation counts where n is the number of sensors in the array (space dimension) and r is the dimension of signal subspace. Real-time implementation of subspace tracking is needed in some applications and regarding that the number of sensors is usually much more than the number of sources ($n \gg r$), algorithms with $O(n^3)$ or even $O(n^2r)$ are not preferred in these cases.

In this paper, we present a recursive algorithm for tracking the signal subspace spanned by the eigenvectors corresponding to the r largest eigenvalues. This algorithm relies on an interpretation of the signal subspace as the solution of a constrained optimization problem based on an approximated projection. The orthonormality of the basis is the constraint which is used in this optimization problem. We will derive both exact and recursive solutions for this problem. We call our approach as constrained projection approximation subspace tracking (CPAST). This algorithm avoids the orthonormalization step in each iteration. We will show that order of computation of the proposed algorithm is $O(nr)$, and thus, it is appropriate for real-time applications.

This paper is organized as follows. In Section 2, the signal mathematical model is presented, and signal and noise subspaces are defined. In Section 3, our approach as a constrained optimization problem is introduced and derivation of the solution is described. Recursive implementations of the proposed solution are derived in Section 4. In Section 5, fast CPAST algorithm with $O(nr)$ complexity is presented. The algorithm used for tracking the signal subspace rank is discussed in Section 6. In Section 7, simulations are used to evaluate the performance of the proposed algorithms and to compare these performances with other existing subspace tracking algorithms. Finally, the main conclusions of this paper are summarized in Section 8.

2. Signal Mathematical Model

Consider the samples $\mathbf{x}(t)$, recorded during the observation time on the n sensor outputs of an array, satisfying the following model:

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

where $\mathbf{x} \in C^n$ is the vector of sensor outputs, $\mathbf{s} \in C^r$ is the vector of complex signal amplitudes, $\mathbf{n} \in C^n$ is an additive noise vector, $\mathbf{A}(\theta) = [\mathbf{a}(\theta_1), \mathbf{a}(\theta_2), \ldots, \mathbf{a}(\theta_r)] \in C^{n\times r}$ is the matrix of the steering vectors $\mathbf{a}(\theta_j)$, and θ_j, $j = 1, 2, \ldots, r$ is the parameter of the jth source, for example, its DOA. It is assumed that $\mathbf{a}(\theta_j)$ is a smooth function of θ_j and that its form is known (i.e., the array is calibrated). We assume that the elements of $\mathbf{s}(t)$ are stationary random processes, and the elements of $\mathbf{n}(t)$ are zero-mean stationary random processes which are uncorrelated with the elements of $\mathbf{s}(t)$. The covariance matrix of the sensors' outputs can be written in the following form:

$$\mathbf{R} = E\left\{\mathbf{x}(t)\,\mathbf{x}^H(t)\right\} = \mathbf{ASA}^H + \mathbf{R}_n, \tag{2}$$

where $\mathbf{S} = E\{\mathbf{s}(t)\mathbf{s}^H(t)\}$ is the signal covariance matrix assumed to be nonsingular ("H" denotes Hermitian transposition), and $\mathbf{R}_n$ is the noise covariance matrix.

Let λ_i and $\mathbf{u}_i$ ($i = 1, 2, \ldots, n$) be the eigenvalues and the corresponding orthonormal eigenvectors of $\mathbf{R}$. In matrix notation, we have $\mathbf{R} = \mathbf{U}\sum\mathbf{U^H}$ with $\sum = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ and $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_n]$, where $\operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix consisting of the diagonal elements λ_i. If we assume that the noise is spatially white with the equal variance σ^2, then the eigenvalues in descending order are given by

$$\lambda_1 \geq \cdots \geq \lambda_r > \lambda_{r+1} = \cdots = \lambda_n = \sigma^2. \tag{3}$$

The dominant eigenpairs $(\lambda_i, \mathbf{u}_i)$ for $i = 1, \ldots, r$ are termed the signal eigenvalues and signal eigenvectors, respectively, while $(\lambda_i, \mathbf{u}_i)$ for $i = r+1, \ldots, n$ are referred to as the noise eigenvalues and noise eigenvectors, respectively. The column spans of

$$\mathbf{U}_S = [\mathbf{u}_1, \ldots, \mathbf{u}_r], \qquad \mathbf{U}_N = [\mathbf{u}_{r+1}, \ldots, \mathbf{u}_n] \tag{4}$$

are called as the signal and noise subspace, respectively. Since the input vector dimension n is often larger than $2r$, it is more efficient to work with the lower dimensional signal subspace than with the noise subspace.

Working with subspaces has some benefits. In the applications that the eigenvalues are not needed, we can apply subspace algorithms which do not estimate eigenvalues and avoid extra computations. In addition, sometimes it is not necessary to know the eigenvectors exactly. For example, in the MUSIC, minimum norm, or ESPRIT algorithms, the use of an arbitrary orthonormal basis of the signal subspace is sufficient. These facts show the reason for the interest in using subspaces in many applications.

3. Constrained Projection Approximation Subspace Tracking

A well-known method for computing the principal subspace of the data is projection approximation subspace tracking (PAST) method. It tracks the dominant subspace of dimension r spanned by the correlation matrix $\mathbf{C}_{xx}$. The columns of signal subspace of PAST method are not exactly orthonormal. The deviation from the orthonormality depends on the signal-to-noise ratio (SNR) and the forgetting factor β. This lack of orthonormality affects seriously the performance of post-processing algorithms which are dependant on orthonormality of the basis. To overcome this problem, we propose the following constrained optimization problem.

Let $\mathbf{x} \in C^n$ be a stationary complex valued random vector process with the autocorrelation matrix $\mathbf{C}_{xx} = E\{\mathbf{xx}^H\}$ which is assumed to be positive definite. We consider the following minimization problem:

$$\begin{aligned}&\underset{\mathbf{W}}{\text{minimize}}\ J'(\mathbf{W}(t)) = \sum_{i=1}^{t}\beta^{t-i}\|\mathbf{x}(i) - \mathbf{W}(t)\mathbf{y}(i)\|^2 \\ &\text{subject to } \mathbf{W}^H(t)\mathbf{W}(t) = \mathbf{I}_r,\end{aligned} \tag{5}$$

where $\mathbf{I}_r$ is the $r \times r$ identity matrix, $\mathbf{y}(t) = \mathbf{W}^H(t-1)\mathbf{x}(t)$ is the r-dimensional compressed data vector, and $\mathbf{W}$ is an $n \times r$ ($r \leq n$) orthonormal subspace basis full rank matrix. Since the above minimization is the PAST cost function, (5) leads to the signal subspace. In addition, the aforementioned constraint guarantees the orthonormality of the signal subspace. The use of the forgetting factor $0 < \beta \leq 1$ is intended to ensure that data in the distant times are downweighted in order to preserve the tracking capability when the system operates in a nonstationary environment.

To solve this constrained problem, we use Lagrange multipliers method. So, after expanding the expression for $J'(\mathbf{W}(t))$, we can replace (5) with the following problem:

$$\begin{aligned}\underset{\mathbf{W}}{\text{minimize}}\ h(\mathbf{W}) = \operatorname{tr}(\mathbf{C}) &- 2\operatorname{tr}\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{x}(i)\mathbf{y}^H(i)\mathbf{W}^H(t)\right) \\ &+ \operatorname{tr}\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{y}(i)\mathbf{y}^H(i)\mathbf{W}^H(t)\mathbf{W}(t)\right) \\ &+ \lambda\left\|\mathbf{W}^H\mathbf{W} - \mathbf{I}_r\right\|_F^2,\end{aligned} \tag{6}$$

where $\operatorname{tr}(\mathbf{C})$ is the trace of the matrix $\mathbf{C}$, $\|\cdot\|_F$ denotes the Frobenius norm, and λ is the Lagrange multiplier. We can rewrite $h(\mathbf{W})$ in the following form:

$$\begin{aligned}h(\mathbf{W}) &= \operatorname{tr}(\mathbf{C}) - 2\operatorname{tr}\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{x}(i)\mathbf{y}^H(i)\mathbf{W}^H(t)\right) \\ &+ \operatorname{tr}\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{y}(i)\mathbf{y}^H(i)\mathbf{W}^H(t)\mathbf{W}(t)\right) \\ &+ \lambda\operatorname{tr}\left(\mathbf{W}^H(t)\mathbf{W}(t)\mathbf{W}^H(t)\mathbf{W}(t) - 2\mathbf{W}^H(t)\mathbf{W}(t) + \mathbf{I}_r\right).\end{aligned} \tag{7}$$

Let $\nabla h = 0$, where ∇ is the gradient operator with respect to $\mathbf{W}$, then we have

$$\begin{aligned}-\sum_{i=1}^{t}\beta^{t-i}\mathbf{x}(i)\mathbf{y}^H(t) + \sum_{i=1}^{t}\beta^{t-i}\mathbf{W}(\mathbf{t})\mathbf{y}(i)\mathbf{y}^H(t) \\ + \lambda\left[-2\mathbf{W}(\mathbf{t}) + 2\mathbf{W}(\mathbf{t})\mathbf{W}^H(\mathbf{t})\mathbf{W}(\mathbf{t})\right] = 0,\end{aligned} \tag{8}$$

which can be rewritten in the following form:

$$\begin{aligned}\mathbf{W}(t) &= \left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{x}(i)\mathbf{y}^H(i)\right) \\ &\times\left[\sum_{i=1}^{t}\beta^{t-i}\mathbf{y}(i)\mathbf{y}^H(i) - 2\lambda\mathbf{I}_r + 2\lambda\mathbf{W}^H(t)\mathbf{W}(t)\right]^{-1}.\end{aligned} \tag{9}$$

If we substitute $\mathbf{W}(t)$ from (9) into the constraint which is $\mathbf{W}^H\mathbf{W} = \mathbf{I}_r$, we obtain

$$\begin{aligned}&\left[\sum_{i=1}^{t}\beta^{t-i}\mathbf{y}(i)\mathbf{y}^H(i) - 2\lambda\mathbf{I}_r + 2\lambda\mathbf{W}^H(t)\mathbf{W}(t)\right]^{-H} \\ &\times\left[\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{y}(i)\mathbf{x}^H(i)\right)\right]\left[\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{x}(i)\mathbf{y}^H(i)\right)\right] \\ &\times\left[\sum_{i=1}^{t}\beta^{t-i}\mathbf{y}(i)\mathbf{y}^H(i) - 2\lambda\mathbf{I}_r + 2\lambda\mathbf{W}^H(t)\mathbf{W}(t)\right]^{-1} \\ &= \mathbf{I}_r.\end{aligned} \tag{10}$$

Now, we define matrix $\mathbf{L}$ as follows:

$$\mathbf{L} = \sum_{i=1}^{t}\beta^{t-i}\mathbf{y}(i)\mathbf{y}^H(i) - 2\lambda\mathbf{I}_r + 2\lambda\mathbf{W}^H(t)\mathbf{W}(t). \tag{11}$$

It follows from (9), (10), and (11) that

$$\mathbf{L}^{-H}\left[\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{y}(i)\mathbf{x}^H(i)\right)\right]\left[\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{x}(i)\mathbf{y}^H(i)\right)\right]\mathbf{L}^{-1} = \mathbf{I}_r. \tag{12}$$

Right and left multiplying (12) by $\mathbf{L}$ and $\mathbf{L}^H$, respectively, and using the fact that $\mathbf{L} = \mathbf{L}^H$, we get

$$\left[\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{y}(i)\mathbf{x}^H(i)\right)\right]\left[\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{x}(i)\mathbf{y}^H(i)\right)\right] = \mathbf{L}^2. \tag{13}$$

It follows from (13) that

$$\begin{aligned}\mathbf{L} &= \left[\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{y}(i)\mathbf{x}^H(i)\right)\left(\sum_{i=1}^{t}\beta^{t-i}\mathbf{x}(i)\mathbf{y}^H(i)\right)\right]^{1/2} \\ &= \left[\mathbf{C}_{xy}^H(t)\mathbf{C}_{xy}(t)\right]^{1/2},\end{aligned} \tag{14}$$

where $(\cdot)^{1/2}$ denotes the square root of a matrix and $\mathbf{C}_{xy}(t)$ is defined as follows:

$$\mathbf{C}_{xy}(t) = \sum_{i=1}^{t}\beta^{t-i}\mathbf{x}(i)\mathbf{y}^H(i). \tag{15}$$

Using (11) and the definition of $\mathbf{C}_{xy}(t)$, we can rewrite (9) in the following form:

$$\mathbf{W}(t) = \mathbf{C}_{xy}(t)\,\mathbf{L}^{-1}. \tag{16}$$

Now, using (14) and (16), we can achieve the following fundamental solution:

$$\mathbf{W}(t) = \mathbf{C}_{xy}(t)\left(\mathbf{C}_{xy}^{H}(t)\,\mathbf{C}_{xy}(t)\right)^{-1/2}. \tag{17}$$

This CPAST algorithm guarantees the orthonormality of the columns of $\mathbf{W}(t)$. It can be seen from (17) that for calculation of the proposed solution just $\mathbf{C}_{xy}(t)$ is needed and calculation of $\mathbf{C}_{xx}(t)$, which is a necessary part of some subspace estimation algorithms, is avoided. Thus, efficient implementation of the proposed solution can reduce the complexity of computations and this is one of the advantages of this solution.

Recursive computation of the $n \times r$ matrix $\mathbf{C}_{xy}(t)$ (by using (15)) requires $O(nr)$ operations. The computation of $\mathbf{W}(t)$ using (17) demands additional $O(nr^2) + O(r^3)$ operations. So, the direct implementation of the CPAST method given by (17) needs $O(nr^2)$ operations.

4. Adaptive CPAST Algorithm

Let us define an $r \times r$ matrix $\mathbf{\Psi}(t)$ which represents the distance between consecutive subspaces as below:

$$\mathbf{\Psi}(t) = \mathbf{W}^{H}(t-1)\,\mathbf{W}(t). \tag{18}$$

Since $\mathbf{W}(t-1)$ approximately spans the dominant subspace of $\mathbf{C}_{xx}(t)$, we have

$$\mathbf{W}(t) \approx \mathbf{W}(t-1)\,\mathbf{\Psi}(t). \tag{19}$$

This is a key step towards obtaining an algorithm for fast subspace tracking using orthogonal iteration. Equations (18) and (19) will be used later.

The $n \times r$ matrix $\mathbf{C}_{xy}(t)$ can be updated recursively in an efficient way which will be discussed in the following sections.

4.1. Recursion for the Correlation Matrix $\mathbf{C_{xx}(t)}$. Let $\mathbf{x}(t)$ be a sequence of n-dimensional data vectors. The correlation matrix $\mathbf{C}_{xx}(t)$, used for signal subspace estimation, can be estimated recursively as follows:

$$\mathbf{C}_{xx}(t) = \sum_{i=1}^{t}\beta^{t-i}\mathbf{x}(i)\,\mathbf{x}^{H}(i) = \beta\mathbf{C}_{xx}(t-1) + \mathbf{x}(t)\,\mathbf{x}^{H}(t), \tag{20}$$

where $0 < \beta < 1$ is the forgetting factor. The windowing method used in (20) is denoted as exponential windowing. Indeed, this kind of windowing tends to smooth the variations of the signal parameters and allows a low complexity update at each time. Thus, it is suitable for slowly changing signals.

For sudden signal parameter changes, the use of a truncated window offers faster tracking. However, subspace trackers based on the truncated window have more computational complexity. In this case, the correlation matrix is estimated in the following way:

$$\begin{aligned}\mathbf{C}_{xx}(t) &= \sum_{i=t-l+1}^{t}\beta^{t-i}\mathbf{x}(i)\,\mathbf{x}^{H}(i)\\ &= \beta\mathbf{C}_{xx}(t-1) + \mathbf{x}(t)\,\mathbf{x}^{H}(t) - \beta^{l}\mathbf{x}(t-l)\,\mathbf{x}^{H}(t-l)\\ &= \beta\mathbf{C}_{xx}(t-1) + \mathbf{z}(t)\,\mathbf{G}\mathbf{z}^{H}(t),\end{aligned} \tag{21}$$

where $l > 0$ is the length of the truncated window, and $\mathbf{z}$ and $\mathbf{G}$ are defined in the following form:

$$\begin{aligned}\mathbf{z}(t) &= \Big[\mathbf{x}(t)\ \vdots\ \mathbf{x}(t-l)\Big]_{n\times 2},\\ \mathbf{G} &= \begin{bmatrix}1 & 0\\ 0 & -\beta^{l}\end{bmatrix}_{2\times 2}.\end{aligned} \tag{22}$$

4.2. Recursion for the Cross Correlation Matrix $\mathbf{C_{xy}(t)}$. To achieve a recursive form for $\mathbf{C}_{xy}(t)$ in the exponential window case, let us use (15), (20), and the definition of $\mathbf{y}(t)$ to derive

$$\begin{aligned}\mathbf{C}_{xy}(t) &= \mathbf{C}_{xx}(t)\,\mathbf{W}(t-1)\\ &= \beta\mathbf{C}_{xx}(t-1)\,\mathbf{W}(t-1) + \mathbf{x}(t)\,\mathbf{y}^{H}(t).\end{aligned} \tag{23}$$

By applying projection approximation (19) at time $t-1$, (23) can be rewritten in the following form:

$$\begin{aligned}\mathbf{C}_{xy}(t) &\approx \beta\mathbf{C}_{xx}(t-1)\,\mathbf{W}(t-2)\,\mathbf{\Psi}(t-1) + \mathbf{x}(t)\,\mathbf{y}^{H}(t)\\ &= \beta\mathbf{C}_{xy}(t-1)\,\mathbf{\Psi}(t-1) + \mathbf{x}(t)\,\mathbf{y}^{H}(t).\end{aligned} \tag{24}$$

In the truncated window case, the recursion can be obtained in a similar way. To this end, by using (21), employing projection approximation, and doing some manipulations, we get

$$\mathbf{C}_{xy}(t) = \beta\mathbf{C}_{xy}(t-1)\,\mathbf{\Psi}(t-1) + \mathbf{z}(t)\,\mathbf{G}\hat{\mathbf{z}}^{H}(t), \tag{25}$$

where

$$\hat{\mathbf{z}}(t) = \Big[\mathbf{y}(t)\ \vdots\ \mathbf{W}^{H}(t-1)\,\mathbf{x}(t-l)\Big]_{n\times 2}. \tag{26}$$

4.3. Recursion for Signal Subspace $\mathbf{W(t)}$. Now, we want to find a recursion for fast update of signal subspace. Let us use (14) to rewrite (16) as below

$$\mathbf{W}(t) = \mathbf{C}_{xy}(t)\,\mathbf{\Phi}(t), \tag{27}$$

where

$$\mathbf{\Phi}(t) = \left[\mathbf{C}_{xy}^{H}(t)\,\mathbf{C}_{xy}(t)\right]^{-1/2}. \tag{28}$$

Substituting (27) into (24) and right multiplying by $\mathbf{\Phi}(t)$, results the following recursion:

$$\begin{aligned}\mathbf{W}(t) \approx{}& \beta\mathbf{W}(t-1)\,\mathbf{\Phi}^{-1}(t-1)\,\mathbf{\Psi}(t-1)\,\mathbf{\Phi}(t)\\ &+ \mathbf{x}(t)\,\mathbf{y}^{H}(t)\,\mathbf{\Phi}(t).\end{aligned} \tag{29}$$

Now, left multiplying (29) by $\mathbf{W}^H(t-1)$, right multiplying it by $\mathbf{\Phi}^{-1}(t)$, and using (18), we obtain

$$\mathbf{\Psi}(t)\mathbf{\Phi}^{-1}(t) \approx \beta\mathbf{\Phi}^{-1}(t-1)\mathbf{\Psi}(t-1) + \mathbf{y}(t)\mathbf{y}^H(t). \tag{30}$$

To further reduce the complexity, we apply the matrix inversion lemma to (30). The matrix inversion lemma can be written as follows:

$$(\mathbf{A}+\mathbf{BCD})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{DA}^{-1}\mathbf{B}+\mathbf{C}^{-1}\right)^{-1}\mathbf{DA}^{-1}. \tag{31}$$

Using matrix inversion lemma, we can replace (30) with the following equation:

$$\left[\mathbf{\Psi}(t)\mathbf{\Phi}^{-1}(t)\right]^{-1} = \frac{1}{\beta}\mathbf{\Psi}^{-1}(t-1)\mathbf{\Phi}(t-1)\left[\mathbf{I}_r - \mathbf{y}(t)\mathbf{g}(t)\right], \tag{32}$$

where

$$\mathbf{g}(t) = \frac{\mathbf{y}^H(t)\mathbf{\Psi}^{-1}(t-1)\mathbf{\Phi}(t-1)}{\beta + \mathbf{y}^H(t)\mathbf{\Psi}^{-1}(t-1)\mathbf{\Phi}(t-1)\mathbf{y}(t)}. \tag{33}$$

Now, left multiplying (32) by $\mathbf{\Phi}^{-1}(t)$ leads to the following recursion:

$$\mathbf{\Psi}^{-1}(t) = \frac{1}{\beta}\mathbf{\Phi}^{-1}(t)\mathbf{\Psi}^{-1}(t-1)\mathbf{\Phi}(t-1)\left[\mathbf{I}_r - \mathbf{y}(t)\mathbf{g}(t)\right]. \tag{34}$$

Finally, by taking an inverse from both sides of (34), the following recursion is obtained for $\mathbf{\Psi}(t)$:

$$\mathbf{\Psi}(t) = \beta\left[\mathbf{I}_r - \mathbf{y}(t)\mathbf{g}(t)\right]^{-1}\mathbf{\Phi}^{-1}(t-1)\mathbf{\Psi}(t-1)\mathbf{\Phi}(t). \tag{35}$$

It is straightforward to show that for the truncated window case, the recursions for $\mathbf{W}(t)$ and $\mathbf{\Psi}(t)$ are as follows:

$$\begin{aligned}\mathbf{W}(t) &= \beta\mathbf{W}(t-1)\mathbf{\Phi}^{-1}(t-1)\mathbf{\Psi}(t-1)\mathbf{\Phi}(t)\\ &\quad + \mathbf{z}(t)\mathbf{G}\hat{\mathbf{z}}^H(t)\mathbf{\Phi}(t),\\ \mathbf{\Psi}(t) &= \beta\left(\mathbf{I}_r - \hat{\mathbf{z}}(t)\mathbf{v}^H(t)\right)^{-1}\mathbf{\Phi}^{-1}(t-1)\mathbf{\Psi}(t-1)\mathbf{\Phi}(t),\end{aligned} \tag{36}$$

where

$$\begin{aligned}\mathbf{v}(t) &= \frac{1}{\beta}\mathbf{\Phi}^H(t-1)\mathbf{\Psi}^{-H}(t-1)\hat{\mathbf{z}}(t)\\ &\quad\times\left[\mathbf{G}^{-1} + \frac{1}{\beta}\hat{\mathbf{z}}^H(t)\mathbf{\Psi}^{-1}(t-1)\mathbf{\Phi}(t-1)\hat{\mathbf{z}}(t)\right]^{-H}.\end{aligned} \tag{37}$$

Using (24) and (28), an efficient algorithm for updating $\mathbf{\Phi}(t)$ in the exponential window case can be obtained. It is as follows:

$$\begin{aligned}\alpha &= \mathbf{x}^H(t)\mathbf{x}(t),\\ \mathbf{U}(t) &= \beta\mathbf{\Psi}^H(t-1)\left(\mathbf{C}_{xy}^H(t-1)\mathbf{x}(t)\right)\mathbf{y}^H(t),\end{aligned} \tag{38}$$

$$\begin{aligned}\mathbf{\Omega}(t) &= \mathbf{C}_{xy}^H(t)\mathbf{C}_{xy}(t)\\ &= \beta^2\mathbf{\Psi}^H(t-1)\mathbf{\Omega}(t-1)\mathbf{\Psi}(t-1)\\ &\quad + \mathbf{U}(t) + \mathbf{U}^H(t) + \alpha\mathbf{y}(t)\mathbf{y}^H(t),\end{aligned} \tag{39}$$

$$\mathbf{\Phi}(t) = \mathbf{\Omega}^{-1/2}(t). \tag{40}$$

Similarly, it can be shown that an efficient recursion for truncated window case is as follows:

$$\begin{aligned}\mathbf{U}(t) &= \beta\mathbf{\Psi}^H(t-1)\left(\mathbf{C}_{xy}^H(t-1)\mathbf{z}(t)\right)\mathbf{G}\hat{\mathbf{z}}^H(t),\\ \mathbf{\Omega}(t) &= \beta^2\mathbf{\Psi}^H(t-1)\mathbf{\Omega}(t-1)\mathbf{\Psi}(t-1) + \mathbf{U}(t)\\ &\quad + \mathbf{U}^H(t) + \hat{\mathbf{z}}(t)\mathbf{G}^H\left(\mathbf{z}^H(t)\mathbf{z}(t)\right)\mathbf{G}\hat{\mathbf{z}}^H(t),\\ \mathbf{\Phi}(t) &= \mathbf{\Omega}^{-1/2}(t).\end{aligned} \tag{41}$$

The pseudocodes of the exponential window CPAST algorithm and the truncated window CPAST algorithm are presented in Tables 1 and 2, respectively.

5. Fast CPAST Algorithm

The subspace tracker in CPAST can be considered a fast algorithm because it requires only a single nr^2 operation count in the computation of the matrix product $\mathbf{W}(t-1)(\mathbf{\Phi}^{-1}(t-1)\mathbf{\Psi}(t-1)\mathbf{\Phi}(t))$ in (29). However, in this section, we further reduce the complexity of the CPAST algorithm.

By employing (34), then (29) can be replaced with the following recursion:

$$\begin{aligned}\mathbf{W}(t) &= \mathbf{W}(t-1)\left(\mathbf{I}_r - \mathbf{y}(t)\mathbf{g}^H(t)\right)\mathbf{\Psi}(t)\\ &\quad + \mathbf{x}(t)\mathbf{y}^H(t)\mathbf{\Phi}(t).\end{aligned} \tag{42}$$

Further simplification and complexity reduction comes from an inspection of $\mathbf{\Psi}(t)$. This matrix represents the distance between consecutive subspaces. When the forgetting factor is relatively close to 1, this distance will be small and $\mathbf{\Psi}(t)$ will approach to the identity matrix. Our simulation results approve this claim. So, we use the approximation $\mathbf{\Psi}(t) = \mathbf{I}_r$ to simplify the signal subspace recursion as follows:

$$\begin{aligned}\mathbf{W}(t) &= \mathbf{W}(t-1) - \left(\mathbf{W}(t-1)\mathbf{y}(t)\right)\mathbf{g}^H(t)\\ &\quad + \mathbf{x}(t)\mathbf{y}^H(t)\mathbf{\Phi}(t).\end{aligned} \tag{43}$$

To further reduce the complexity, we substitute $\mathbf{\Psi}(t) = \mathbf{I}_r$ in (30) and apply the matrix inversion lemma to it. The result is as follows:

$$\mathbf{\Phi}(t) = \frac{1}{\beta}\mathbf{\Phi}(t-1)\left(\mathbf{I}_r - \frac{\mathbf{y}(t)\mathbf{f}^H(t)}{\mathbf{f}^H(t)\mathbf{y}(t)+\beta}\right), \tag{44}$$

Table 1: Exponential window CPAST algorithm.

The algorithm	Cost (MAC count)
$\mathbf{W}(0) = \begin{bmatrix} \mathbf{I} \\ \cdots \\ 0 \end{bmatrix}$; $\mathbf{C}_{xy}(0) = \begin{bmatrix} \mathbf{I} \\ \cdots \\ 0 \end{bmatrix}$; $\boldsymbol{\Phi}(0) = \boldsymbol{\Omega}(0) = \boldsymbol{\Psi}(0) = \mathbf{I}_r$	
FOR $t = 1, 2, \ldots$ DO	
$\mathbf{y}(t) = \mathbf{W}^H(t-1)\mathbf{x}(t)$	nr
$\mathbf{C}_{xy}(t) = \beta\mathbf{C}_{xy}(t-1)\boldsymbol{\Psi}(t-1) + \mathbf{x}(t)\mathbf{y}^H(t)$	$2nr$
$\mathbf{U}(t) = \beta(\mathbf{C}_{xy}^H(t-1)\mathbf{x}(t))\mathbf{y}^H(t)$	$nr + r^2$
$\boldsymbol{\Omega}(t) = \beta^2\boldsymbol{\Omega}(t-1) + \mathbf{U}(t) + \mathbf{U}^H(t) + \mathbf{y}(t)(\mathbf{x}^H(t)\mathbf{x}(t))\mathbf{y}^H(t)$	$n + O(r^2)$
$\boldsymbol{\Phi}(t) = \boldsymbol{\Omega}^{-1/2}(t)$	$O(r^3)$
$\mathbf{W}(t) = \mathbf{W}(t-1)(\beta\boldsymbol{\Phi}^{-1}(t-1)\boldsymbol{\Psi}(t-1)\boldsymbol{\Phi}(t)) + \mathbf{x}(t)(\mathbf{y}^H(t)\boldsymbol{\Phi}(t))$	$nr^2 + nr + O(r^2)$
$\mathbf{g}(t) = \dfrac{\mathbf{y}^H(t)\boldsymbol{\Psi}^{-1}(t-1)\boldsymbol{\Phi}(t-1)}{\beta + \mathbf{y}^H(t)\boldsymbol{\Psi}^{-1}(t-1)\boldsymbol{\Phi}(t-1)\mathbf{y}(t)}$	$O(r^2)$
$\boldsymbol{\Psi}(t) = \beta(\mathbf{I}_r - \mathbf{y}(t)\mathbf{g}(t))^{-1}\boldsymbol{\Phi}^{-1}(t-1)\boldsymbol{\Psi}(t-1)\boldsymbol{\Phi}(t)$	$O(r^2)$

Table 2: Truncated window CPAST algorithm.

The algorithm
$\mathbf{W}(0) = \begin{bmatrix} \mathbf{I} \\ \cdots \\ 0 \end{bmatrix}$; $\mathbf{C}_{xy}(0) = \begin{bmatrix} \mathbf{I} \\ \cdots \\ 0 \end{bmatrix}$; $\boldsymbol{\Phi}(0) = \boldsymbol{\Omega}(0) = \boldsymbol{\Psi}(0) = \mathbf{I}_r$
$\mathbf{G} = \begin{bmatrix} 1 & 0 \\ 0 & -\beta^l \end{bmatrix}_{2\times 2}$
FOR $t = 1, 2, \ldots$ DO
$\mathbf{y}(t) = \mathbf{W}^H(t-1)\mathbf{x}(t)$
$\mathbf{z}(t) = \begin{bmatrix} \mathbf{x}(t) & \vdots & \mathbf{x}(t-l) \end{bmatrix}_{n\times 2}$
$\hat{\mathbf{z}}(t) = \begin{bmatrix} \mathbf{y}(t) & \vdots & \mathbf{W}^H(t-1)\mathbf{x}(t-l) \end{bmatrix}_{r\times 2}$
$\mathbf{C}_{xy}(t) = \beta\mathbf{C}_{xy}(t-1)\boldsymbol{\Psi}(t-1) + \mathbf{z}(t)\mathbf{G}\hat{\mathbf{z}}^H(t)$
$\mathbf{U}(t) = \beta\boldsymbol{\Psi}^H(t-1)(\mathbf{C}_{xy}^H(t-1)\mathbf{z}(t))\mathbf{G}\hat{\mathbf{z}}^H(t)$
$\boldsymbol{\Omega}(t) = \beta^2\boldsymbol{\Psi}^H(t-1)\boldsymbol{\Omega}(t-1)\boldsymbol{\Psi}(t-1) + \mathbf{U}(t) + \mathbf{U}^H(t) + \hat{\mathbf{z}}(t)\mathbf{G}^H(\mathbf{z}^H(t)\mathbf{z}(t))\mathbf{G}\hat{\mathbf{z}}^H(t)$.
$\boldsymbol{\Phi}(t) = \boldsymbol{\Omega}^{-1/2}(t)$
$\mathbf{W}(t) = \beta\mathbf{W}(t-1)\boldsymbol{\Phi}^{-1}(t-1)\boldsymbol{\Psi}(t-1)\boldsymbol{\Phi}(t) + \mathbf{z}(t)\mathbf{G}\hat{\mathbf{z}}^H(t)\boldsymbol{\Phi}(t)$
$\mathbf{v}(t) = \dfrac{1}{\beta}\boldsymbol{\Phi}^H(t-1)\boldsymbol{\Psi}^{-H}(t-1)\hat{\mathbf{z}}(t) \times [\mathbf{G}^{-1} + \dfrac{1}{\beta}\hat{\mathbf{z}}^H(t)\boldsymbol{\Psi}^{-1}(t-1)\boldsymbol{\Phi}(t-1)\hat{\mathbf{z}}(t)]^{-H}$
$\boldsymbol{\Psi}(t) = \beta(\mathbf{I}_r - \hat{\mathbf{z}}(t)\mathbf{v}^H(t))^{-1}\boldsymbol{\Phi}^{-1}(t-1)\boldsymbol{\Psi}(t-1)\boldsymbol{\Phi}(t)$

where

$$\mathbf{f}(t) = \boldsymbol{\Phi}^H(t-1)\,\mathbf{y}(t). \tag{45}$$

In a similar way, it can be shown easily that using $\boldsymbol{\Psi}(t) = \mathbf{I}_r$ for the truncated window case, yields the following recursions:

$$\begin{aligned} \mathbf{W}(t) &= \mathbf{W}(t-1) - (\mathbf{W}(t-1)\,\hat{\mathbf{z}}(t))\,\mathbf{v}^H(t) \\ &\quad + \mathbf{z}(t)\,\mathbf{G}\hat{\mathbf{z}}^H(t)\,\boldsymbol{\Phi}(t), \\ \boldsymbol{\Phi}(t) &= \frac{1}{\beta}\boldsymbol{\Phi}(t-1)\left[\mathbf{I}_r - \hat{\mathbf{z}}(t)\,\mathbf{v}^H(t)\right], \end{aligned} \tag{46}$$

where

$$\mathbf{v}(t) = \frac{1}{\beta}\boldsymbol{\Phi}^H(t-1)\,\hat{\mathbf{z}}(t)\left[\mathbf{G}^{-1} + \frac{1}{\beta}\hat{\mathbf{z}}^H(t)\,\boldsymbol{\Phi}(t-1)\,\hat{\mathbf{z}}(t)\right]^{-H}. \tag{47}$$

The above simplification reduces the computational complexity of the CPAST algorithm to $O(nr)$. So, we name this simplified CPAST algorithm as fast CPAST. The pseudocodes for exponential window and truncated window versions of fast CPAST are presented in Tables 3 and 4, respectively.

6. Fast Signal Subspace Rank Tracking

Most of subspace tracking algorithms just can track the dominant subspace and they need to know the signal subspace dimension before they begin to track. However, the proposed fast CPAST can track the dimension of the signal subspace. For example, when this algorithm is used for DOA estimation, it can estimate and track the number of signal sources.

The key idea in estimating the signal subspace dimension is to compare the estimated noise power $\sigma^2(t)$ and the signal eigenvalues. The number of eigenvalues which are greater than the noise power can be used as an estimate of signal

Table 3: Exponential window fast CPAST algorithm.

The algorithm	Cost (MAC count)
$\mathbf{W}(0) = \begin{bmatrix} \mathbf{I} \\ \cdots \\ 0 \end{bmatrix}$; $\mathbf{\Phi}(0) = \mathbf{\Omega}(0) = \mathbf{\Psi}(0) = \mathbf{I}$	
FOR $t = 1, 2, \ldots$ DO	
$\mathbf{y}(t) = \mathbf{W}^H(t-1)\mathbf{x}(t)$	nr
$\mathbf{f}(t) = \mathbf{\Phi}^H(t-1)\mathbf{y}(t)$	r^2
$\mathbf{g}(t) = \dfrac{\mathbf{y}^H(t)\mathbf{\Phi}(t-1)}{\beta + \mathbf{y}^H(t)\mathbf{\Phi}(t-1)\mathbf{y}(t)}$	r
$\mathbf{\Phi}(t) = \dfrac{1}{\beta}\mathbf{\Phi}(t-1)(\mathbf{I}_r - \dfrac{\mathbf{y}(t)\mathbf{f}^H(t)}{\mathbf{f}^H(t)\mathbf{y}(t)+\beta})$	$3r^2 + r$
$\mathbf{W}(t) = \mathbf{W}(t-1) - (\mathbf{W}(t-1)\mathbf{y}(t))\mathbf{g}(t) + \mathbf{x}(t)(\mathbf{y}^H(t)\mathbf{\Phi}(t))$	$3nr + r^2$

Table 4: Truncated window fast CPAST algorithm.

The algorithm
$\mathbf{W}(0) = \begin{bmatrix} \mathbf{I} \\ \cdots \\ 0 \end{bmatrix}$; $\mathbf{C}_{xy}(0) = \begin{bmatrix} \mathbf{I} \\ \cdots \\ 0 \end{bmatrix}$; $\mathbf{\Phi}(0) = \mathbf{\Omega}(0) = \mathbf{\Psi}(0) = \mathbf{I}$
$\mathbf{G} = \begin{bmatrix} 1 & 0 \\ 0 & -\beta^l \end{bmatrix}_{2\times 2}$
FOR $t = 1, 2, \ldots$ DO
$\mathbf{z}(t) = \begin{bmatrix}\mathbf{x}(t) & \vdots & \mathbf{x}(t-l)\end{bmatrix}_{n\times 2}$
$\mathbf{y}(t) = \mathbf{W}^H(t-1)\mathbf{x}(t)$
$\hat{\mathbf{z}}(t) = \begin{bmatrix}\mathbf{y}(t) & \vdots & \mathbf{W}^H(t-1)\mathbf{x}(t-l)\end{bmatrix}_{r\times 2}$
$\mathbf{v}(t) = \dfrac{1}{\beta}\mathbf{\Phi}^H(t-1)\hat{\mathbf{z}}(t)[\mathbf{G}^{-1} + \dfrac{1}{\beta}\hat{\mathbf{z}}^H(t)\mathbf{\Phi}(t-1)\hat{\mathbf{z}}(t)]^{-H}$
$\mathbf{\Phi}(t) = \dfrac{1}{\beta}\mathbf{\Phi}(t-1)[\mathbf{I}_r - \hat{\mathbf{z}}(t)\mathbf{v}^H(t)]$
$\mathbf{W}(t) = \mathbf{W}(t-1) - (\mathbf{W}(t-1)\hat{\mathbf{z}}(t))\mathbf{v}^H(t) + \mathbf{z}(t)(\mathbf{G}\hat{\mathbf{z}}^H(t)\mathbf{\Phi}(t))$

subspace dimension. Any algorithm which can estimate and track the $\sigma^2(t)$ can be used in the subspace rank tracking algorithm.

Suppose that the input signal can be decomposed as a linear superposition of a signal $\mathbf{s}(t)$ and zero mean white Gaussian noise process $\mathbf{n}(t)$ as follows:

$$\mathbf{x}(t) = \mathbf{s}(t) + \mathbf{n}(t). \tag{48}$$

As the signal and noise are assumed to be independent, we have

$$\mathbf{C}_{xx} = \mathbf{C}_s + \mathbf{C}_n, \tag{49}$$

where $\mathbf{C}_s = E\{\mathbf{s}\mathbf{s}^H\}$ and $\mathbf{C}_n = E\{\mathbf{n}\mathbf{n}^H\} = \sigma^2\mathbf{I}_n$.

We assume that $\mathbf{C}_s$ has at most $r_{\max} < n$ nonvanishing eigenvalues. If r is the exact number of nonzero eigenvalues, we can use EVD to decompose $\mathbf{C}_s$ as below:

$$\begin{aligned}\mathbf{C}_s &= \begin{bmatrix}\mathbf{V}_s^{(r)} & \vdots & \mathbf{V}_s^{(n-r)}\end{bmatrix}\begin{bmatrix}\mathbf{\Lambda}_s^{(r)} & 0 \\ 0 & 0\end{bmatrix}\begin{bmatrix}\mathbf{V}_s^{(r)^H} \\ \cdots \\ \mathbf{V}_s^{(n-r)^H}\end{bmatrix} \\ &= \mathbf{V}_s^{(r)}\mathbf{\Lambda}_s^{(r)}\mathbf{V}_s^{(r)^H}.\end{aligned} \tag{50}$$

It can be shown that the data covariance matrix can be decomposed as follows:

$$\mathbf{C}_{xx} = \mathbf{V}_s^{(r)}\mathbf{\Lambda}_s\mathbf{V}_s^{(r)^H} + \mathbf{V}_n\mathbf{\Lambda}_n\mathbf{V}_n^H, \tag{51}$$

where $\mathbf{V}_n$ denotes the noise subspace. Using (49)–(51), we have

$$\mathbf{V}_s^{(r)}\mathbf{\Lambda}_s\mathbf{V}_s^{(r)^H} + \mathbf{V}_n\mathbf{\Lambda}_n\mathbf{V}_n^H = \mathbf{V}_s^{(r)}\mathbf{\Lambda}_s^{(r)}\mathbf{V}_s^{(r)^H} + \boldsymbol{\sigma}^2\mathbf{I}_n. \tag{52}$$

Since $\mathbf{C}_{xy}(t) = \mathbf{C}_{xx}(t)\mathbf{W}(t-1)$, (39) can be replaced with the following equation:

$$\begin{aligned}&\mathbf{\Omega}(t) \\ &= \mathbf{C}_{xy}^H(t)\,\mathbf{C}_{xy}(t) \\ &= \mathbf{W}^H(t-1)\,\mathbf{C}_{xx}^2(t)\,\mathbf{W}(t-1) \\ &= \mathbf{W}^H(t-1)\left[\mathbf{V}_s^{(r)}(t)\,\mathbf{\Lambda}_s^2(t)\,\mathbf{V}_s^{(r)^H}(t) + \mathbf{V}_n(t)\,\mathbf{\Lambda}_n^2(t)\,\mathbf{V}_n^H(t)\right] \\ &\quad\times \mathbf{W}(t-1).\end{aligned} \tag{53}$$

Using projection approximation and the fact that the dominant eigenvectors of the data and the dominant eigenvectors of the signal are equal, we conclude that $\mathbf{W}(t) = \mathbf{V}_s^{(r)}$. Using this result and the orthogonality of the signal and noise subspaces, we can rewrite (53) in the following way:

$$\begin{aligned}\mathbf{\Omega}(t) &= \mathbf{W}^H(t-1)\,\mathbf{W}(t)\,\mathbf{\Lambda}_s^2(t)\,\mathbf{W}^H(t)\,\mathbf{W}(t-1) \\ &= \mathbf{\Psi}(t)\,\mathbf{\Lambda}_s^2(t)\,\mathbf{\Psi}^H(t).\end{aligned} \tag{54}$$

Table 5: Signal subspace rank estimation.

```
For each time step do
For $k = 1, 2, \ldots, r_{\max}$
  if $\Lambda_s(k,k) > \alpha\sigma^2$
    $\hat{r}(t) = \hat{r}(t) + 1$; increment estimate of number of sources
  end
end
```

Multiplying left and right sides of (52) by $\mathbf{W}^H(t-1)$ and $\mathbf{W}(t-1)$, respectively, we obtain

$$\boldsymbol{\Lambda}_s = \boldsymbol{\Lambda}_s^{(r)} + \sigma^2\mathbf{I}_r. \tag{55}$$

As r is not known, we replace it with $r_{\max}$, and take the traces of both sides of (55). This yields

$$\operatorname{tr}(\boldsymbol{\Lambda}_s) = \operatorname{tr}\left(\boldsymbol{\Lambda}_s^{(r_{\max})}\right) + \sigma^2 r_{\max}. \tag{56}$$

Now, we define the signal power P_s and the data power P_x as follows:

$$P_s = \frac{1}{n}\operatorname{tr}\left(\boldsymbol{\Lambda}_s^{(r_{\max})}\right) = \frac{1}{n}\operatorname{tr}(\boldsymbol{\Lambda}_s) - \frac{r_{\max}}{n}\sigma^2, \tag{57}$$

$$P_x = \frac{1}{n}E\left\{\mathbf{x}^H\mathbf{x}\right\}. \tag{58}$$

An estimator for data power is as follows:

$$P_x(t) = \beta P_x(t-1) + \frac{1}{n}\mathbf{x}^H(t)\,\mathbf{x}(t). \tag{59}$$

Since the signal and noise are statistically independent, it follows from (57) that

$$\sigma^2 = P_x - P_s = P_x - \frac{1}{n}\operatorname{tr}(\boldsymbol{\Lambda}_s) + \frac{r_{\max}}{n}\sigma^2. \tag{60}$$

Solving (60) for σ^2 gives [14]

$$\sigma^2 = \frac{n}{n - r_{\max}}P_x - \frac{1}{n - r_{\max}}\operatorname{tr}(\boldsymbol{\Lambda}_s). \tag{61}$$

The adaptive tracking of the signal subspace rank requires $\boldsymbol{\Lambda}_s$ and the data power at each iteration. $\boldsymbol{\Lambda}_s$ can be obtained by EVD of $\boldsymbol{\Omega}(t)$ and the data power can be obtained using (59) at each iteration. Table 5 summarizes the procedure of signal subspace rank estimation. The parameter α used in this procedure is a constant that its value should be selected. Usually, a value greater than one is selected for α. The advantage of using this procedure for tracking the signal subspace rank is that it has a low computational load.

7. Simulation Results

In this section, we use simulations to demonstrate the applicability and performance of the fast CPAST algorithm and to compare the performance of fast CPAST with other subspace tracking algorithms. To do so, we consider the use of the proposed algorithm in DOA estimation context. Many of DOA estimation algorithms require an estimate of the signal subspace. Once this estimate is obtained, it can be used in the DOA estimation algorithm for finding the desired DOA's. So, we investigate the performance of fast CPAST in estimating the signal subspace and compare it with other subspace tracking algorithms.

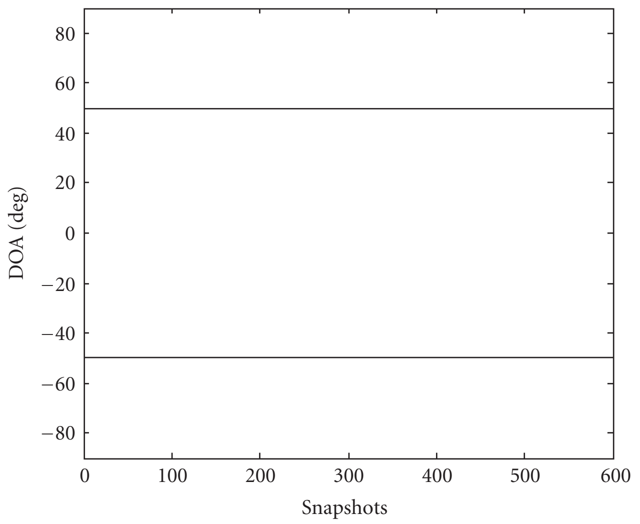


Figure 1: The trajectories of sources in the first simulation scenario.

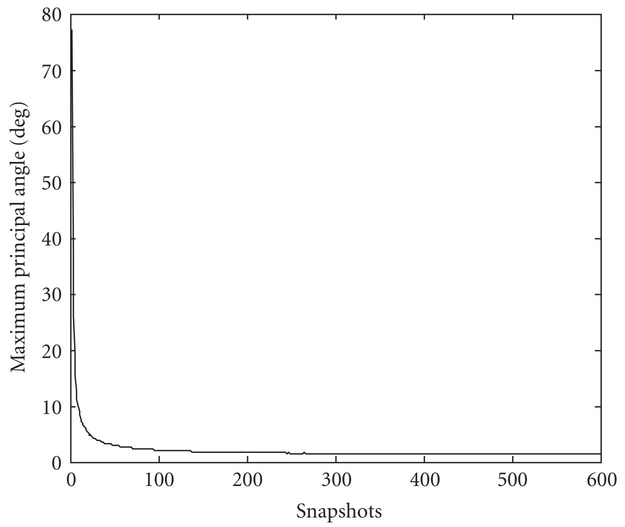
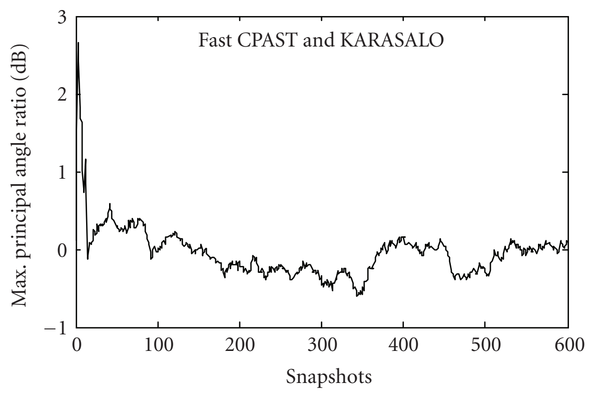
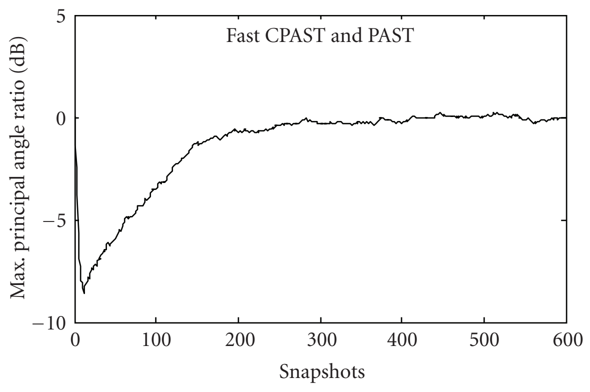


Figure 2: Maximum principal angle of the fast CPAST algorithm in the first simulation scenario.

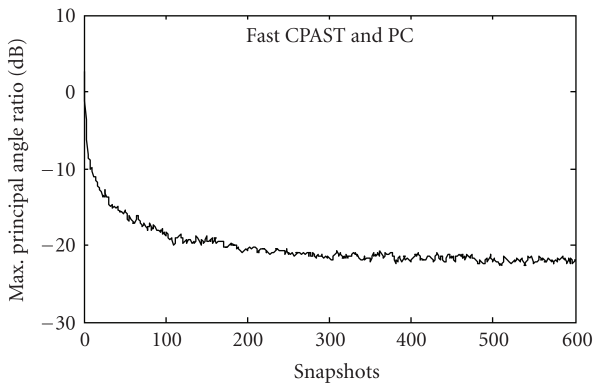
The subspace tracking algorithms used in our simulations and their complexities are shown in Table 6. The Karasalo [1] algorithm is based on subspace averaging. OPAST is the orthonormal version of PAST proposed by Abed-Meriam et al. [13]. The BISVD algorithms are introduced by Strobach [14] and are based on bi-iteration. PROTEUS and PC are the algorithms developed by Champagne and Liu [15, 16] and are based on perturbation theory. NIC is based on a novel information criterion proposed by Miao and Hua [12]. API and FAPI which are based on power



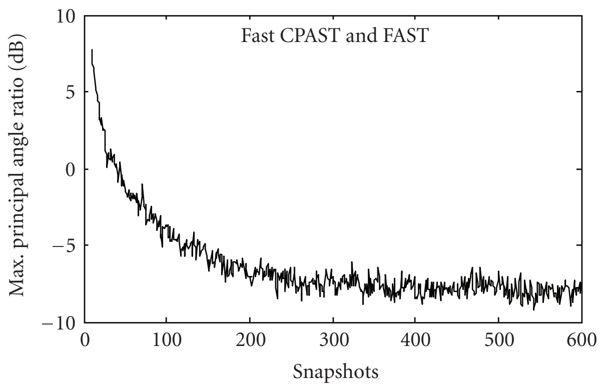
(a)



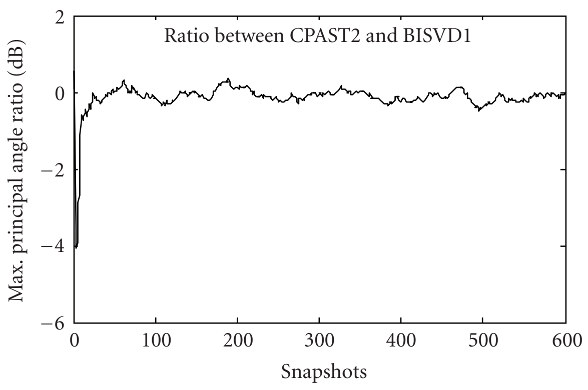
(b)



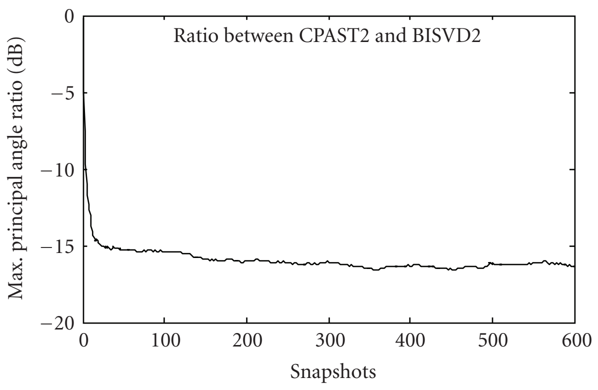
(c)



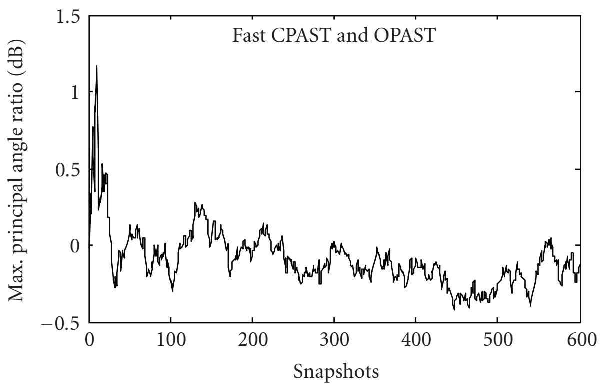
(d)



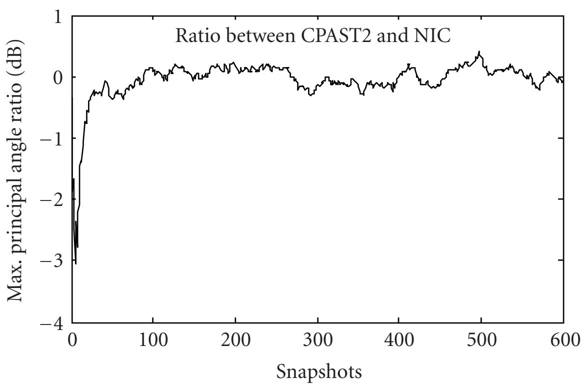
(e)



(f)

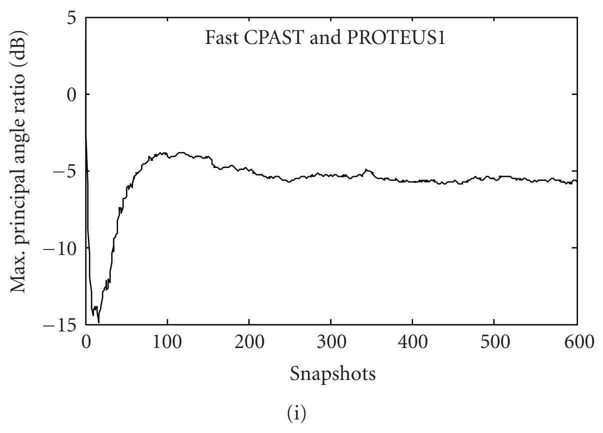


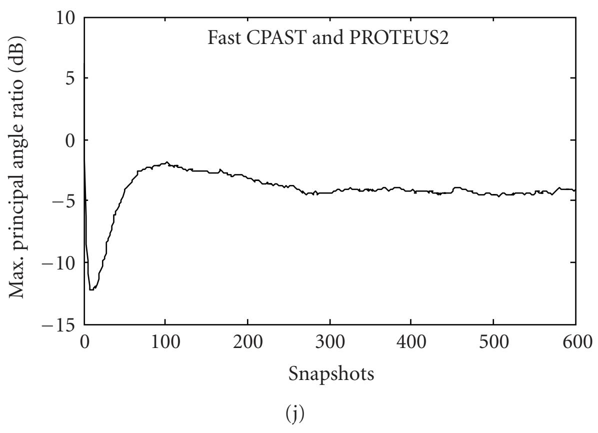
(g)

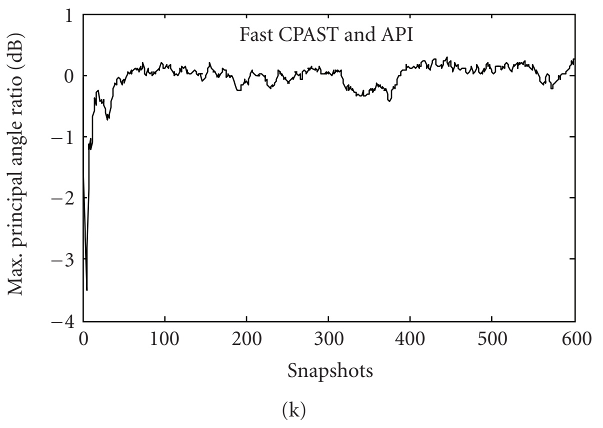


(h)

Figure 3: Continued.







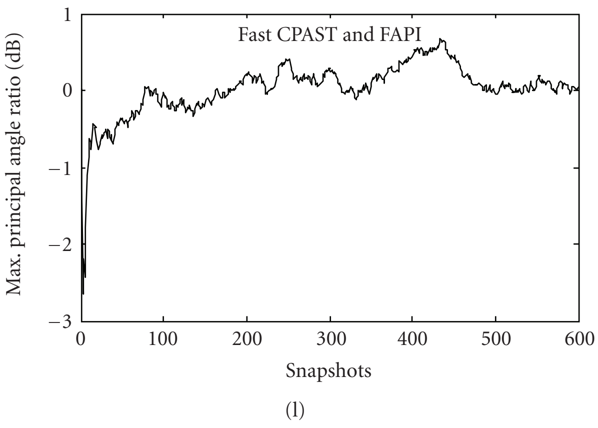


Figure 3: Ratio of maximum principal angles of fast CPAST and other algorithms in the first simulation scenario.

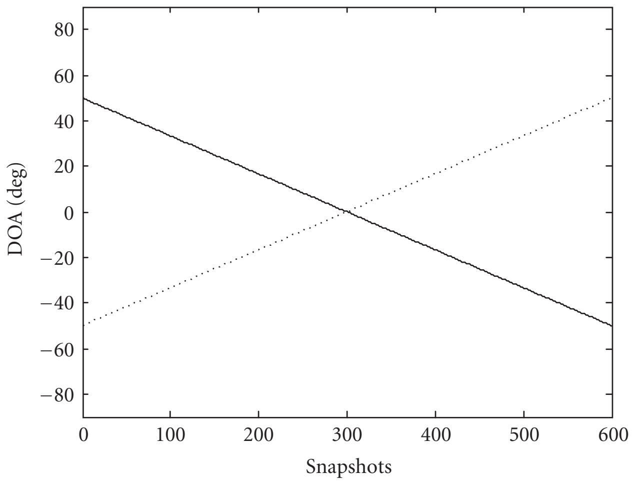


Figure 4: The trajectories of sources in the second simulation scenario.

Table 6: Subspace tracking algorithms used in the simulations and their complexities.

Algorithm	Cost (MAC count)
Fast CPAST	$4nr + 2r + 5r^2$
KARASALO	$nr^2 + 3nr + 2n + O(r^2) + O(r^3)$
PAST	$3nr + 2r^2 + O(r)$
BISVD1	$nr^2 + 3nr + 2n + O(r^2) + O(r^3)$
BISVD2	$4nr + 2n + O(r^2) + O(r^3)$
OPAST	$4nr + n + 2r^2 + O(r)$
NIC	$5nr + O(r) + O(r^2)$
PROTEUS1	$(3/4)nr^2 + (15/4)nr + O(n) + O(r) + O(r^2)$
PROTEUS2	$(21/4)nr + O(n) + O(r) + O(r^2)$
API	$nr^2 + 3nr + n + O(r^2) + O(r^3)$
FAPI	$3nr + 2n + 5r^2 + O(r^3)$
PC	$5nr + O(n)$
FAST	$Nr^2 + 10nr + 2n + 64 + O(r^2) + O(r^3)$

iteration are introduced by Badeau et al. [17, 18]. The FAST algorithm is proposed by Real et al. [19].

In the following subsections the performance of the fast CPAST algorithm is investigated using simulations. In Section 7.1, the performance of fast CPAST is compared with the algorithms mentioned in Table 6 in several cases. In Section 7.2, effect of nonstationarity and the parameters n and SNR on the performance of the fast CPAST algorithm is investigated. In Section 7.3, the performance of the proposed signal subspace rank estimator is investigated. In Section 7.4, the case that we have an abrupt change in the signal DOA is considered and the performance of the proposed fast CPAST

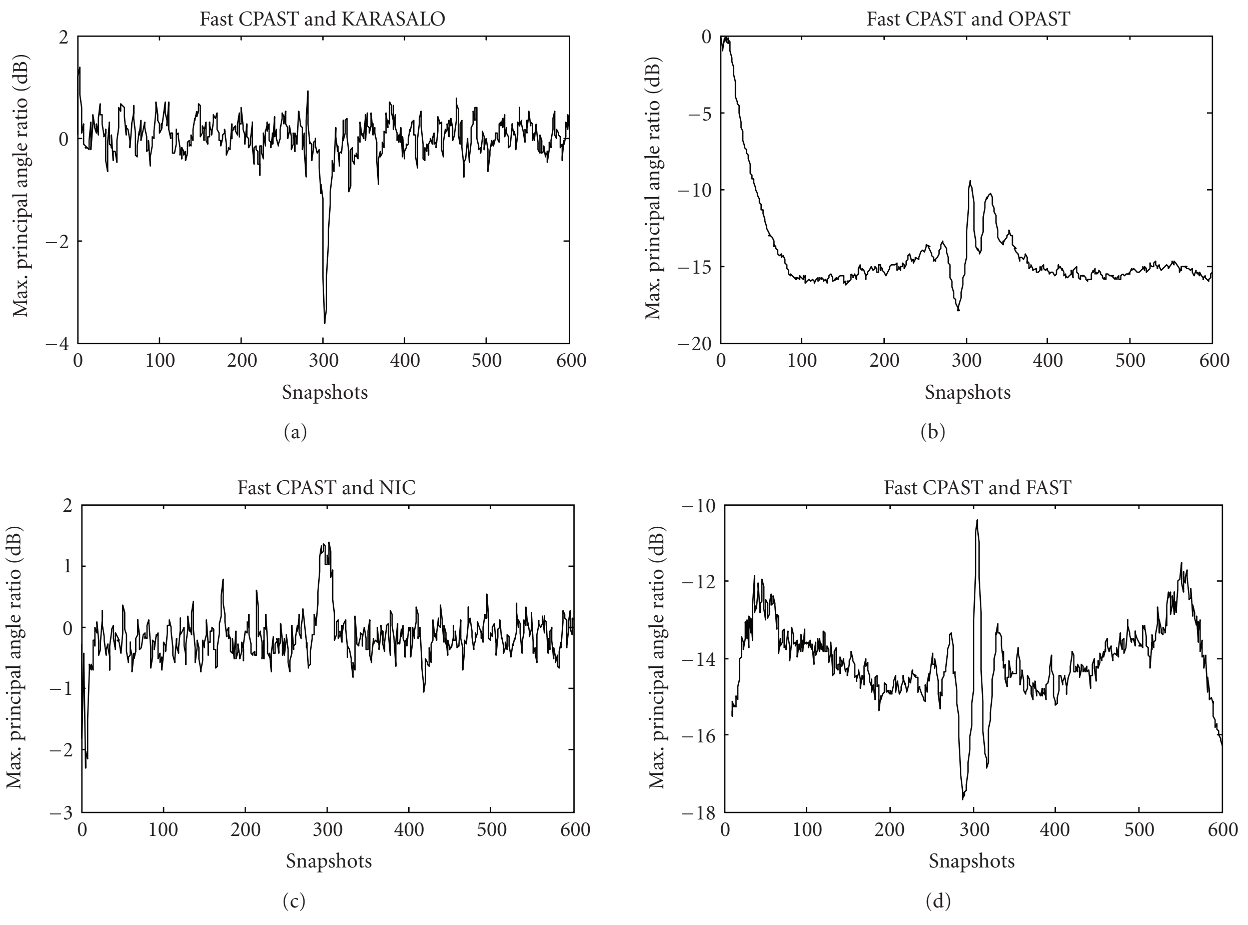


Figure 5: Ratio of maximum principal angles of fast CPAST and several other algorithms in the second simulation scenario.

algorithm with truncated window is compared with that of fast CPAST algorithm with exponential window.

In all simulations of this subsection, we have used the Monte Carlo simulation and the number of simulation runs used for obtaining each point is equal to 100. The only exceptions are Section 7.3 and part 4 in Section 7.2 where the results are obtained using one simulation run.

7.1. Comparison of the Performance of Fast CPAST with That of Other Algorithms. In this subsection, we consider a uniform linear array where the number of sensors is $n = 17$ and the distance between adjacent sensors is equal to half wavelength. In each scenario, an appropriate value is selected for the forgetting factor. In stationary case, old data could be useful. So, large values of forgetting factor ($\beta = 0.99$) are used. On the other hand, in nonstationary scenario where old data are not reliable, smaller values ($\beta = 0.75$) are used. Generally, the value selected for the forgetting factor should depend on the variation of data and improper choosing of forgetting factor can degrade the performance of the algorithm.

In the first scenario, the test signal is the sum of signals of two sources plus a white Gaussian noise. The SNR of each source is equal to 10 dB. Figure 1 shows the trajectories of these sources. Since this scenario describes a stationary case, a forgetting factor of $\beta = 0.99$ has been selected.

Figure 2 shows the maximum principal angle of fast CPAST algorithm in each snapshot. Principal angles [20] are measures of the difference between the estimated and real subspaces. The principal angles are zero if the compared subspaces are identical. In Figure 3, the maximum principal angle of fast CPAST is compared with other subspace tracking algorithms. In comparisons, the ratio of the maximum principal angles of fast CPAST and the other algorithms are obtained in decibels using the following relation:

$$20 \log \left(\frac{\theta_{\text{CPAST}}}{\theta_{\text{alg}}} \right), \tag{62}$$

where θ_{CPAST} and θ_{alg} denote the maximum principal angles of the fast CPAST and any of the algorithms mentioned in Table 3, respectively. This figure shows that the performance of the fast CPAST is much better than the PC, FAST, BISVD2, PROTEUS1, and PROTEUS2 after the convergence of algorithms. In addition, it can be seen from this figure that the fast CPAST has faster convergence rate than the PAST, BISVD1, NIC, API, and FAPI algorithms.

In the second scenario, we want to investigate the behavior of the fast CPAST in comparison with other

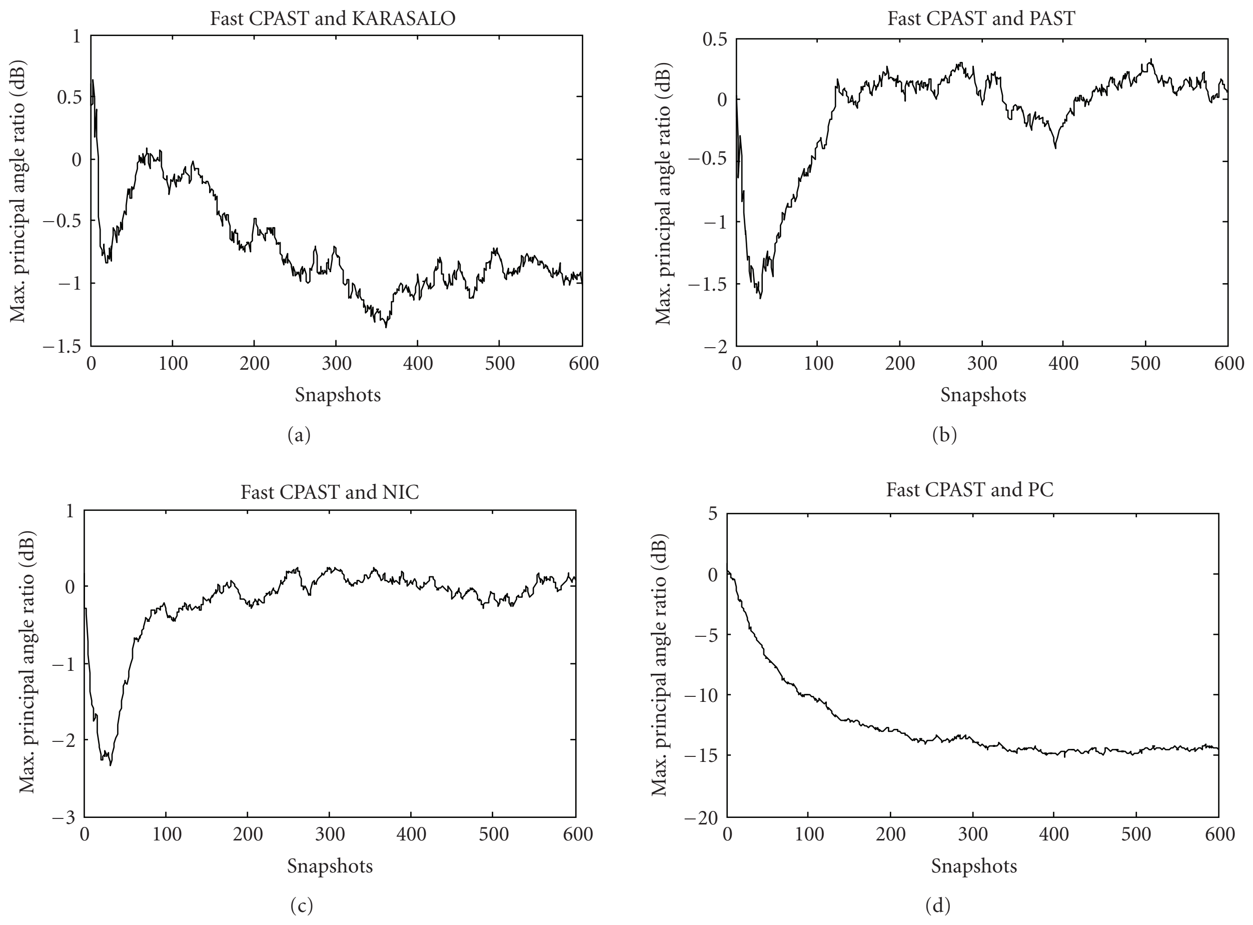


Figure 6: Ratio of maximum principal angles of fast CPAST and several other algorithms in the third simulation scenario.

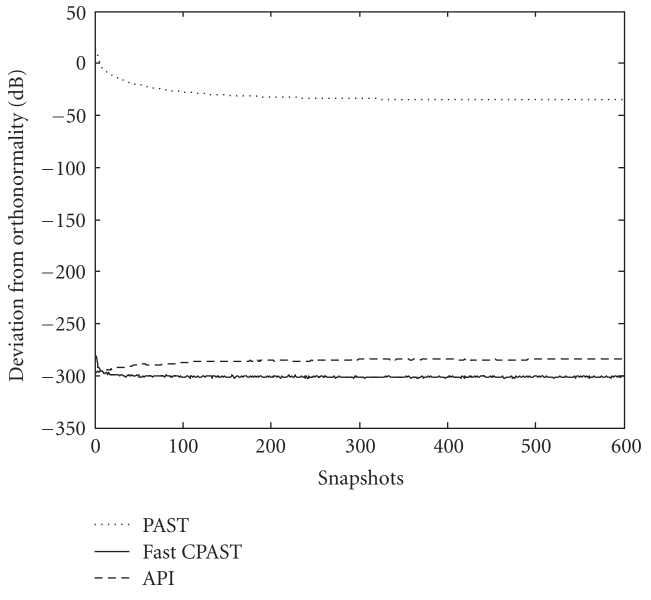


Figure 7: Deviation from orthonormality for three algorithms in the third scenario.

algorithms in a nonstationary environment. The test signal is the sum of signals of two sources plus a white Gaussian noise. Figure 4 shows the trajectories of the sources. Because of the nonstationarity of the environment, the forgetting factor is chosen as $\beta = 0.75$. The SNR of each source is equal to 10 dB. The simulation results showed that the performance of fast CPAST in this scenario is better than most of the other algorithms mentioned in Table 3 and approximately the same as few of them. In Figure 5, the ratio of the maximum principal angle of fast CPAST and some of these algorithms are shown in dB.

In the third scenario, we consider two sources that are stationary and are located at $[-5^\circ, 5^\circ]$. The SNR of each of them is equal to -5 dB. In this scenario β was equal to 0.99. The simulation results showed that the performance of fast CPAST in this scenario is better than some of the other algorithms mentioned in Table 3 after convergence. For other algorithms of Table 3, fast CPAST has a faster convergence, but the performances are similar after the convergence. In Figure 6, the ratio of the maximum principal angle of fast CPAST and some of these algorithms are shown in dB.

Figure 3 through Figure 6 show that the fast CPAST outperforms OPAST, BISVD2, BISVD1, NIC, PROTEUS2, FAPI, and PC algorithms in all three scenarios. In fact, in comparison with algorithms that have a computational complexity equal to $O(nr^2)$ or $O(nr)$, fast CPAST has equal or better performance in all three scenarios.

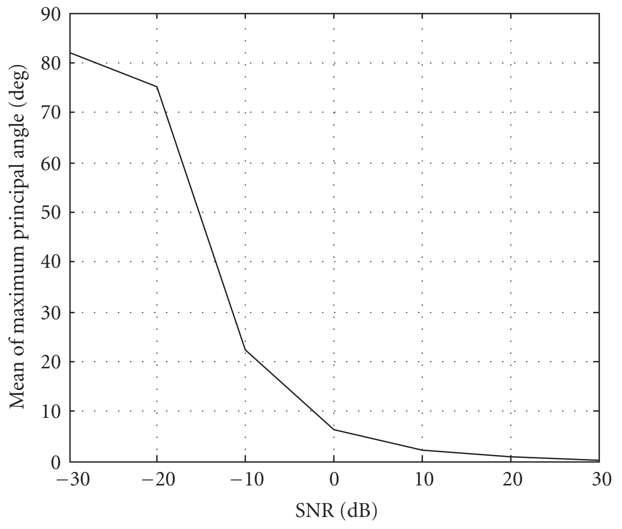


Figure 8: Mean of maximum principal angle versus SNR for two stationary sources located at $[-50^\circ,\ 50^\circ]$.

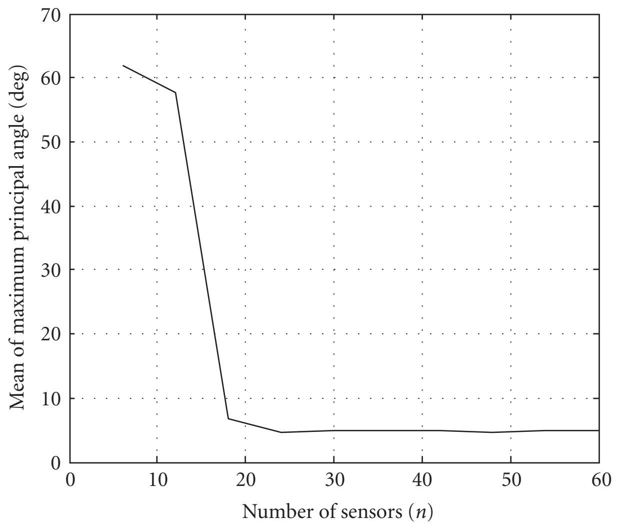


Figure 9: Mean of maximum principal angle versus number of sensors for five stationary sources located at $[-10^\circ,\ -5^\circ,\ 0^\circ,\ 5^\circ,\ 10^\circ]$.

The deviation of the subspace weighting matrix $\mathbf{W}(t)$ from orthonormality can be measured by means of the following error criterion [18]:

$$20\log\left(\left\|\mathbf{W}^H(t)\,\mathbf{W}(t)-\mathbf{I}_r\right\|_F\right). \tag{63}$$

We consider the third scenario for investigating the deviation of the subspace weighting matrix from orthonormality. Table 7 shows the average of orthonormality error given by (63) for algorithms in Table 6. It can be seen from Table 7 that fast CPAST, KARASALO, OPAST, BISVD1, PROTEUS2, FAPI, FAST, API, and PROTEUS1 outperform the other algorithms. In addition, a plot of variations of the orthonormality error with time (snapshot number) is provided in Figure 7 for the fast CPAST, API, and PAST algorithms. The results for other algorithms are not presented here to keep the presentation as concise as possible.

Table 7: Average of orthonormality error given by (63) for algorithms mentioned in Table 6 in the third simulation scenario.

Algorithm	Orthonormality error
fast CPAST, KARASALO, OPAST, BISVD1,	about −300 dB
PROTEUS2, FAPI, FAST	about −300 dB
API	about −285 dB
PROTEUS1	about −265 dB
PAST, NIC	about −30 dB
PC	about 0 dB
BISVD2	about 30 dB

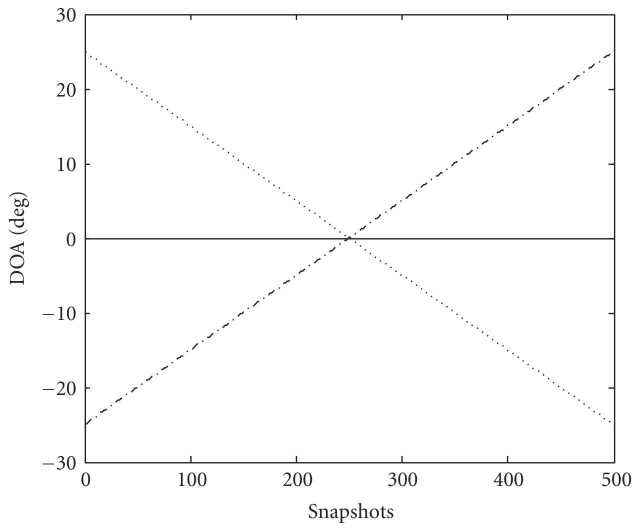


Figure 10: Real trajectories of three crossing-over targets versus number of snapshots.

7.2. Effect of SNR, **n**, *and Nonstationarity on the Performance of the Fast CPAST Algorithm.* In this section, we consider a uniform linear array where the number of sensors is $n = 17$ and the distance between adjacent sensors is equal to half wavelength. The exceptions are Sections 7.2.2 and 7.2.3 where we change the number of sensors.

7.2.1. Effect of the SNR. In this part of Section 7.2, we investigate the influence of SNR on the performance of the fast CPAST algorithm. We consider two sources that are stationary and are located at $[-50^\circ,\ 50^\circ]$. The performance is evaluated for SNR's from −30 dB to 30 dB. Figure 8 shows the mean of maximum principal angle for each SNR. Simulations using fast CPAST and MUSIC showed that for a SNR of −10 dB a mean square error of about 1 degree can be reached in DOA estimation.

7.2.2. Effect of the Number of Sensors. In this part, the effect of increasing number of sensors on the performance of fast

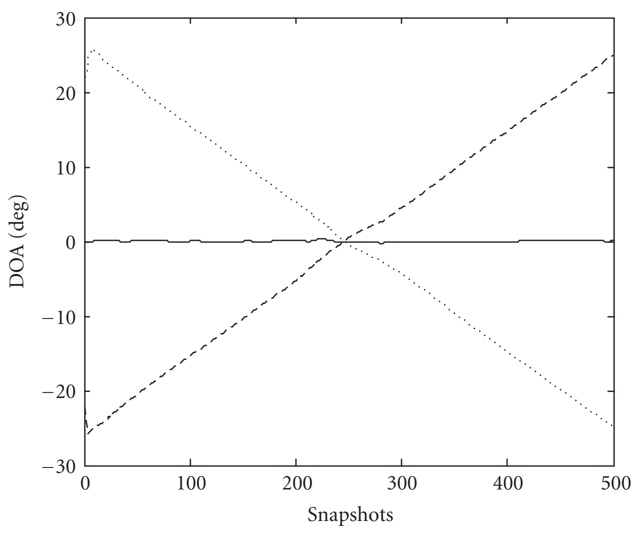


Figure 11: Estimated trajectories of the three crossing-over targets versus number of snapshots.

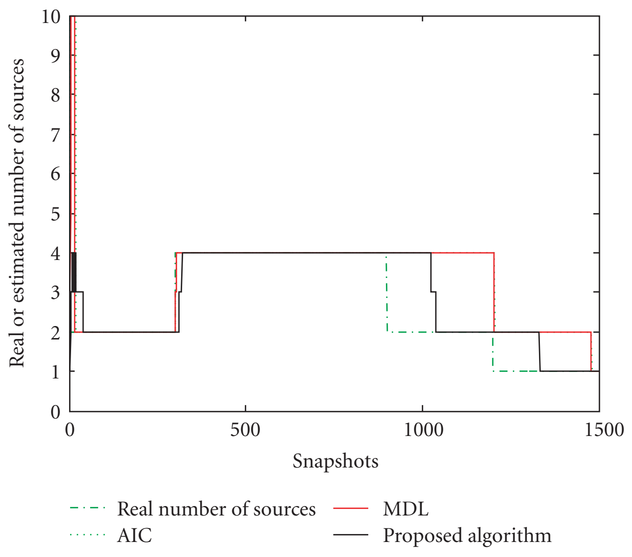


Figure 12: Real and Estimated number of sources versus number of snapshots.

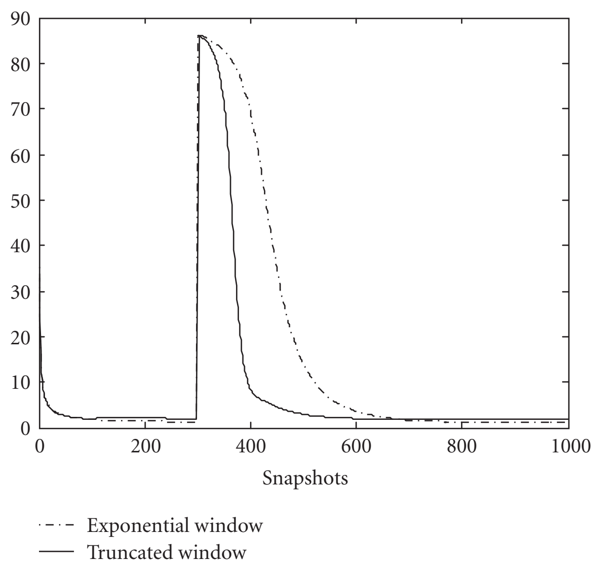


Figure 13: Maximum principal angle of the fast CPAST algorithm with exponential and truncated windows.

CPAST is investigated. To do so, we consider five sources that are stationary and are located at $[-10^\circ, -5^\circ, 0^\circ, 5^\circ, 10^\circ]$ and their SNR is 5 dB. Figure 9 shows the mean of maximum principal angle for $n \in \{6, 7, \ldots, 60\}$. It can be seen that the subspace estimation algorithm reaches its best performance for $n \geq 18$ and it remains approximately unchanged with increasing n.

7.2.3. Effect of a Nonstationary Environment. In this part, we use MUSIC algorithm for finding the DOA's of signal sources impinging on an array of sensors. Let $\{\mathbf{s}_i\}_{i=1}^n$ denote the orthonormal eigenvectors of covariance matrix $\mathbf{R}$. We assume that the corresponding eigenvalues of $\mathbf{R}$ are sorted in descending order. We know that the MUSIC method gives consistent estimates of the DOA's as the minimizing arguments of the following cost function:

$$f_{\mathrm{MUSIC}}(\theta) = a^H(\theta)\left(\mathbf{I}_n - \mathbf{S}\mathbf{S}^H\right)a(\theta), \tag{64}$$

where $\mathbf{S}$ is any orthonormal basis of the signal subspace like $\mathbf{S} = (\mathbf{s}_1, \ldots, \mathbf{s}_r)$, and $a(\theta)$ is the steering vector corresponding to the angle θ.

To demonstrate the capability of the proposed algorithm in target tracking in nonstationary environments, we consider three targets which have crossover in their trajectories. The trajectories of these three targets are depicted in Figure 10. The SNR for each of the three targets is equal to 0 dB and the number of sensors is 17. We have used the fast CPAST algorithm for tracking the signal subspace of these targets, the MUSIC algorithm for estimating their DOA's, and the Kalman filter for tracking their trajectories. The simulation result is shown in Figure 11. It can be seen from Figures 9 and 10 that the combination of fast CPAST, MUSIC, and Kalman filter algorithms has been successful in estimating and tracking the trajectories of the sources.

7.3. Performance of the Proposed Signal Subspace Rank Estimator. In this subsection, we investigate the performance of the proposed signal subspace rank estimator. To this end, we consider the case that we have two sources at first, and then two other sources are added to them at 300th snapshot. Then, at 900th snapshot two of these sources are removed. Finally, at 1200th snapshot, another one of the sources is removed. In this simulation, we assume that $\alpha = 2.5$ and $r_{\max} = 6$. Figure 12 shows the performance of AIC, MDL, and the proposed algorithm in tracking number of sources. It can be seen that the proposed algorithm is successful in tracking the number of sources. In addition, when the number of sources decreases, the proposed rank estimator can track the changes in the number of sources faster than AIC and MDL.

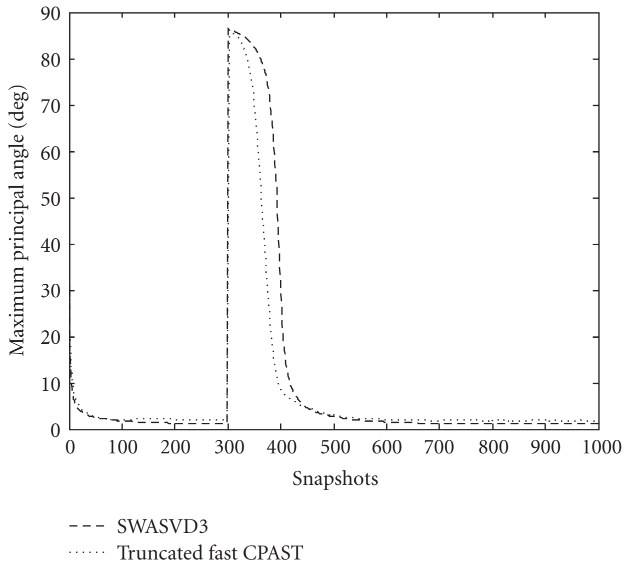


Figure 14: Maximum principal angle of the truncated fast CPAST and SWASVD3 algorithms.

7.4. Performance of the Proposed Fast CPAST Algorithm with Truncated Window. In this section, we compare the convergence behavior of the CPAST algorithm with exponential and truncated windows. We consider a source whose DOA is equal to 10° until 300th snapshot and it changes abruptly to 70° at this snapshot. We assume that the SNR is 10 dB and the forgetting factor is equal to 0.99. Figure 13 shows the maximum principal angle of the CPAST algorithm with exponential and truncated windows. It shows that, in this case, the CPAST algorithm with truncated window and an equivalent window length $l = 1/(1 - \beta)$, converges much faster than the exponential window algorithm.

In order to do more investigation about the performance of truncated fast CPAST algorithm, we have compared its performance with that of SWASVD3 [21] algorithm which uses a truncated window for signal subspace tracking. The scenario that is used in this performance comparison is the same as that of Figure 13 and the length of window is equal to 100 for both algorithms. Figure 14 depicts the result and it can be seen from this figure that the performance of the truncated fast CPAST is superior to that of SWASVD3.

8. Concluding Remarks

In this paper, we introduced an interpretation of the signal subspace as the solution of a constrained optimization problem. We derived the solution of this problem and discussed the applicability of the so-called CPAST algorithm for tracking the subspace. In addition, we derived two recursive formulations of this solution for adaptive implementation. This solution and its recursive implementations avoid the orthonormalization of basis in each update. The computational complexity of one of these algorithms (fast CPAST) is $O(nr)$ which is appropriate for online implementation. The proposed algorithms are efficiently applicable in those post processing applications which need an orthonormal basis for the signal subspace.

In order to compare the performance of the proposed fast CPAST algorithm with other subspace tracking algorithms, several simulation scenarios were considered. The simulation results showed that the performance of fast CPAST is usually better than or at least similar to that of other algorithms. In a second set of simulations, effect of SNR, space dimension n, and nonstationarity on the performance of fast CPAST was investigated. The simulation results showed good performance of fast CPAST with low SNR and nonstationary environment.

References

[1] I. Karasalo, "Estimating the covariance matrix by signal subspace averaging," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 1, pp. 8–12, 1986.

[2] D. J. Rabideau, "Fast, rank adaptive subspace tracking and applications," *IEEE Transactions on Signal Processing*, vol. 44, no. 9, pp. 2229–2244, 1996.

[3] E. M. Dowling, L. P. Ammann, and R. D. DeGroat, "A TQR-iteration based adaptive SVD for real time angle and frequency tracking," *IEEE Transactions on Signal Processing*, vol. 42, no. 4, pp. 914–926, 1994.

[4] M. Moonen, P. Van Dooren, and J. Vandewalle, "A singular value decomposition updating algorithm for subspace tracking," *SIAM Journal on Matrix Analysis and Applications*, vol. 13, no. 4, pp. 1015–1038, 1992.

[5] Y. Hua, Y. Xiang, T. Chen, K. Abed-Meraim, and Y. Miao, "A new look at the power method for fast subspace tracking," *Digital Signal Processing*, vol. 9, no. 4, pp. 297–314, 1999.

[6] C. H. Bischof and G. M. Shroff, "On updating signal subspaces," *IEEE Transactions on Signal Processing*, vol. 40, no. 1, pp. 96–105, 1992.

[7] G. W. Stewart, "An updating algorithm for subspace tracking," *IEEE Transactions on Signal Processing*, vol. 40, no. 6, pp. 1535–1541, 1992.

[8] G. Xu, H. Zha, G. H. Golub, and T. Kailath, "Fast algorithms for updating signal subspaces," *IEEE Transactions on Circuits and Systems II*, vol. 41, no. 8, pp. 537–549, 1994.

[9] J. R. Bunch, C. P. Nielsen, and D. C. Sorensen, "Rank-one modification of the symmetric eigenproblem," *Numerische Mathematik*, vol. 31, no. 1, pp. 31–48, 1978.

[10] R. D. DeGroat, "Noniterative subspace tracking," *IEEE Transactions on Signal Processing*, vol. 40, no. 3, pp. 571–577, 1992.

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

[12] Y. Miao and Y. Hua, "Fast subspace tracking and neural network learning by a novel information criterion," *IEEE Transactions on Signal Processing*, vol. 46, no. 7, pp. 1967–1979, 1998.

[13] K. Abed-Meraim, A. Chkeif, and Y. Hua, "Fast orthonormal PAST algorithm," *IEEE Signal Processing Letters*, vol. 7, no. 3, pp. 60–62, 2000.

[14] P. Strobach, "Bi-iteration SVD subspace tracking algorithms," *IEEE Transactions on Signal Processing*, vol. 45, no. 5, pp. 1222–1240, 1997.

[15] B. Champagne and Q.-G. Liu, "Plane rotation-based EVD updating schemes for efficient subspace tracking," *IEEE Transactions on Signal Processing*, vol. 46, no. 7, pp. 1886–1900, 1998.

[16] B. Champagne, "Adaptive eigendecomposition of data covariance matrices based on first-order perturbations," *IEEE Transactions on Signal Processing*, vol. 42, no. 10, pp. 2758–2770, 1994.

[17] R. Badeau, B. David, and G. Richard, "Fast approximated power iteration subspace tracking," *IEEE Transactions on Signal Processing*, vol. 53, no. 8, pp. 2931–2941, 2005.

[18] R. Badeau, G. Richard, and B. David, "Approximated power iterations for fast subspace tracking," in *Proceedings of the 7th International Symposium on Signal Processing and Its Applications (ISSPA '03)*, vol. 2, pp. 583–586, Paris, France, July 2003.

[19] E. C. Real, D. W. Tufts, and J. W. Cooley, "Two algorithms for fast approximate subspace tracking," *IEEE Transactions on Signal Processing*, vol. 47, no. 7, pp. 1936–1945, 1999.

[20] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, Md, USA, 2nd edition, 1989.

[21] R. Badeau, G. Richard, and B. David, "Sliding window adaptive SVD algorithms," *IEEE Transactions on Signal Processing*, vol. 52, no. 1, pp. 1–10, 2004.