Fast Approximated Power Iteration Subspace Tracking

Roland Badeau, Bertrand David, Gael Richard

Fast Approximated Power Iteration Subspace Tracking

Roland Badeau, *Member, IEEE*, Bertrand David, and Gaël Richard, *Member, IEEE*

***Abstract*— This paper introduces a fast implementation of the power iteration method for subspace tracking, based on an approximation less restrictive than the well known *projection approximation*. This algorithm, referred to as the fast API method, guarantees the orthonormality of the subspace weighting matrix at each iteration. Moreover, it outperforms many subspace trackers related to the power iteration method, such as PAST, NIC, NP3 and OPAST, while having the same computational complexity. The API method is designed for both exponential windows and sliding windows. Our numerical simulations show that sliding windows offer a faster tracking response to abrupt signal variations.**

***Index Terms*— Subspace tracking, projection approximation, power iteration.**

I. Introduction

THE interest in subspace-based methods stems from the fact that they consist in splitting the observations into a set of desired and a set of disturbing components, which can be viewed in terms of signal and noise subspaces. These methods have applications in numerous domains including the fields of adaptive filtering, source localization, or parameter estimation [1]. The estimation of the signal subspace is commonly based on the traditional eigenvalue decomposition (EVD) or singular value decomposition (SVD). However, the main drawback of these decompositions is their inherent complexity. Therefore, there is a real need for fast subspace tracking techniques in the context of adaptive signal processing.

Due to this interest, a large number of approaches have already been introduced. A reference method is I. Karasalo's algorithm [2], which involves the full SVD of a small matrix. A fast tracking method based on Givens rotations (the FST algorithm) is proposed in [3]. Other approaches consist in interlacing a recursive update of the estimated correlation matrix or the data matrix with one or a few steps of a standard SVD or power iteration algorithm. This is the case of the Jacobi SVD method [4], the transposed QR-iteration [5], the orthogonal / bi-orthogonal iteration [6], [7], and the power method [8]. Other matrix decompositions have also successfully been used in subspace tracking (for example the rank-revealing QR factorization [9], the rank-revealing URV decomposition [10], and the Lankzos (bi)-diagonalization [11]). Other techniques rely on the noise and signal subspace averaging method [12], the maximum likelihood principle [13], the operator restriction analysis [14], or the perturbation theory [15].

Roland Badeau, Bertrand David and Gaël Richard are with the Department of Signal and Image Processing, Ecole Nationale Supérieure des Télécommunications (ENST), Paris, France. E-mail: [roland.badeau, bertrand.david, gael.richard]@enst.fr.

The estimation of the signal subspace can also be viewed as a constrained or unconstrained optimization problem [16]–[21], for which the introduction of a projection approximation hypothesis lead to fast subspace tracking methods (see *e.g.* the PAST [22] and NIC [23] algorithms). In [8], it is shown that these subspace trackers are closely linked to the classical power iteration method [24]. Several implementations of this method based on QR factorizations are proposed in [6], among which the Loraf2 and Loraf3 algorithms. However, compared to PAST and NIC, Loraf2 is more computationally demanding and the performance of Loraf3 is degraded. Another fast implementation of the power iteration method, the NP3 algorithm which relies on rank-one matrix updates, is proposed in [8], but our numerical simulations showed that this algorithm does not converge in many situations. An orthonormal version of the PAST algorithm, proposed in [25], can be seen as a fast implementation of the power method and outperforms PAST, NIC and NP3. Concurrently, the recent API method [26], based on the power iteration method and on a new projection approximation, has the same computational complexity as the above mentioned algorithms but provides a better estimation of the dominant subspace.

All these adaptive techniques are designed for exponential windows. Indeed, this choice tends to smooth the variations of the signal parameters, and thus allows a low-complexity update at each time step. However, it is only suitable for slowly changing signals. Conversely, a few subspace trackers are based on sliding windows, which generally require more computations, but offer a faster tracking response to sudden signal changes [22], [27]–[30]. In particular, a sliding window version of the API algorithm is proposed in [31].

This paper presents several fast implementations of the API method. These algorithms present several advantages:

- they can be applied either on an infinite *exponential window* or on a *truncated window* (*e.g.* a sliding window which may have an exponential decrease),
- an orthonormal subspace basis is computed *at each time step*, which is required for some subspace-based estimation methods, such as MUSIC [32],
- they rely on a new projection approximation, less restrictive than the classical one, which leads to better tracking results. In particular, it is shown that the PAST and OPAST subspace trackers can be viewed as approximations of the fast API method.

The paper is organized as follows. In section II, we introduce the various window shapes applied to the data. In section III, the classical power iteration method is reviewed, then the projection approximation is discussed in section IV.

Our approximated power iteration (API) method is introduced in section V, and a fast implementation of this algorithm is proposed in section VI. In section VII, it is shown that both PAST and OPAST can be seen as approximations of the fast API algorithm. In section VIII, the performance of this method is compared to that of several subspace trackers, among which PAST and OPAST. Finally, the main conclusions of this paper are summarized in section IX.

II. Data windowing

Let $\{\boldsymbol{x}(t)\}_{t\in\mathbb{Z}}$ be a sequence of n-dimensional data vectors. We are interested in computing the dominant subspace spanned by its correlation matrix. This matrix can be estimated according to the nature of the data window.

A. Exponential window

The estimated $n \times n$ correlation matrix is defined as

$$\boldsymbol{C}_{xx}(t) = \sum_{u=-\infty}^{t} \beta^{t-u}\, \boldsymbol{x}(u)\, \boldsymbol{x}(u)^H$$

where $0 < \beta < 1$ is the forgetting factor. It can be recursively updated according to the following scheme:

$$\boldsymbol{C}_{xx}(t) = \beta\, \boldsymbol{C}_{xx}(t-1) + \boldsymbol{x}(t)\, \boldsymbol{x}(t)^H. \tag{1}$$

B. Truncated window

The $n \times n$ correlation matrix $\boldsymbol{C}_{xx}(t)$ is estimated on a window of length $l \in \mathbb{N}^*$:

$$\boldsymbol{C}_{xx}(t) = \sum_{u=t-l+1}^{t} \beta^{t-u}\, \boldsymbol{x}(u)\, \boldsymbol{x}(u)^H. \tag{2}$$

where $0 < \beta \leq 1$. The case $\beta = 1$ corresponds to a rectangular (or sliding) window. This matrix can be recursively updated according to the following scheme:

$$\boldsymbol{C}_{xx}(t) = \beta\, \boldsymbol{C}_{xx}(t-1) + \boldsymbol{x}(t)\, \boldsymbol{x}(t)^H - \beta^l \boldsymbol{x}(t-l)\, \boldsymbol{x}(t-l)^H. \tag{3}$$

C. Unified formalization

Both equations (1) and (3) can be written in the form

$$\boldsymbol{C}_{xx}(t) = \beta\, \boldsymbol{C}_{xx}(t-1) + \underline{\boldsymbol{x}}(t)\, \underline{J}\, \underline{\boldsymbol{x}}(t)^H \tag{4}$$

where $\underline{\boldsymbol{x}}(t)$ and $\underline{J}$ are defined according to the window shape:

- in the exponential window case:

$$\underline{J} = 1 \tag{5}$$
$$\underline{\boldsymbol{x}}(t) = \boldsymbol{x}(t) \tag{6}$$

- in the truncated window case:

$$\underline{J} = \begin{bmatrix} 1 & 0 \\ 0 & -\beta^l \end{bmatrix} \tag{7}$$
$$\underline{\boldsymbol{x}}(t) = \left[\, \boldsymbol{x}(t) \;\middle|\; \boldsymbol{x}(t-l) \,\right]. \tag{8}$$

Let p be the rank of the update involved in equation (4). Since $p = 1$ in the exponential window case and $p = 2$ in the truncated window case, p characterizes the window shape. In particular, $\underline{\boldsymbol{x}}(t)$ is a $n \times p$ matrix and $\underline{J}$ is a $p \times p$ matrix.

III. The classical power iteration method

The power iteration method [8] tracks the dominant subspace[1] of dimension $r \leq n$ spanned by the $n \times n$ matrix $\boldsymbol{C}_{xx}(t)$. At each time step, a basis of this subspace is computed, represented by an orthonormal matrix $\boldsymbol{W}(t)$ of dimension $n \times r$. The computation of $\boldsymbol{W}(t)$ consists of a data compression step (9) and an orthonormalization step (10) of the compressed matrix at each iteration:

$$\boldsymbol{C}_{xy}(t) = \boldsymbol{C}_{xx}(t)\, \boldsymbol{W}(t-1) \tag{9}$$

$$\boldsymbol{W}(t)\, \boldsymbol{R}(t) = \boldsymbol{C}_{xy}(t). \tag{10}$$

where $\boldsymbol{C}_{xy}(t)$ can be considered as a $n \times r$ correlation matrix between the n-dimensional data vectors $\boldsymbol{x}(t)$ and the r-dimensional compressed data vectors

$$\boldsymbol{y}(t) = \boldsymbol{W}(t-1)^H \boldsymbol{x}(t). \tag{11}$$

The orthonormalization step (10) involves a $r \times r$ matrix $\boldsymbol{R}(t)$, such that $\boldsymbol{R}(t)^H\, \boldsymbol{R}(t) = \boldsymbol{\Phi}(t)$, where $\boldsymbol{\Phi}(t)$ is the $r \times r$ positive definite matrix $\boldsymbol{C}_{xy}(t)^H \boldsymbol{C}_{xy}(t)$. Consequently, $\boldsymbol{R}(t)^H$ is a square root of $\boldsymbol{\Phi}(t)$. In particular, $\boldsymbol{R}(t)^H$ is equal to the positive definite square root of $\boldsymbol{\Phi}(t)$, right multiplied by a unitary matrix[2]. For example, $\boldsymbol{R}(t)$ can be triangular [6], or positive definite [8].

If $\boldsymbol{C}_{xx}(t)$ remains constant and if its first r eigenvalues are strictly larger than the $(n-r)^{\text{th}}$ others, the power iteration method converges globally and exponentially to the principal subspace [8] [24, pp. 410-411]. Note that the multiplication in step (9) involves $n^2 r$ operations, and the orthonormalization step (10) requires $O(nr^2)$ operations[3]. Because of its high computational cost, this algorithm is not suitable for real-time processing.

IV. The projection approximation

We are now looking for an approximation that will allow us to reduce the complexity. Suppose that $\boldsymbol{W}(t-1)$ *exactly* spans the r-dimensional dominant subspace of $\boldsymbol{C}_{xx}(t)$. Then equation (9) yields

$$\boldsymbol{C}_{xy}(t) = \boldsymbol{W}(t-1)\, \boldsymbol{C}_{yy}(t) \tag{12}$$

where the matrix $\boldsymbol{C}_{yy}(t) \triangleq \boldsymbol{W}(t-1)^H \boldsymbol{C}_{xx}(t)\, \boldsymbol{W}(t-1)$ can be seen as the correlation matrix of the compressed data

[1] The r-dimensional dominant subspace of the positive semidefinite matrix $\boldsymbol{C}_{xx}(t)$ is the subspace spanned by the r eigenvectors of $\boldsymbol{C}_{xx}(t)$ associated to the r eigenvalues of highest magnitude (which are supposed to be strictly greater than the $n-r$ others).

[2] If $\boldsymbol{T}$ is a positive definite matrix, a *square root* of $\boldsymbol{T}$ is any matrix $\boldsymbol{S}$ of the same dimension such that $\boldsymbol{S}\, \boldsymbol{S}^H = \boldsymbol{T}$. Such a matrix is denoted $\boldsymbol{S} = \boldsymbol{T}^{\frac{1}{2}}$. There is only one positive definite square root of $\boldsymbol{T}$. The other square roots are obtained by right multiplying this positive definite square root by any unitary transform. The notation $\boldsymbol{S}^{\frac{1}{2}}$ can denote any of them.

[3] In this paper, operations counts are expressed in terms of multiply / accumulate (MAC) operations, herein referred to as *flops*. Whenever a specific matrix function is used, such as orthonormalization, inversion or square rooting, only the order of the operations count is presented, since the exact operations count depends on the way this function is implemented. Nevertheless, r is supposed to be much lower than n, so that the dominant cost of the power iteration method is that of the first step, whose exact operations count is known ($n^2 r$).

vectors. In this case, $\boldsymbol{W}(t)$ and $\boldsymbol{W}(t-1)$ are two orthonormal matrices spanning the range space of $\boldsymbol{C}_{xy}(t)$, thus

$$\boldsymbol{W}(t) = \boldsymbol{W}(t-1)\,\boldsymbol{\Theta}(t) \tag{13}$$

where $\boldsymbol{\Theta}(t) \triangleq \boldsymbol{W}(t-1)^H \boldsymbol{W}(t)$ is a $r \times r$ orthonormal matrix. Substituting equation (12) into equation (10) and left multiplying by $\boldsymbol{W}(t)^H$ yields the polar decomposition of $\boldsymbol{R}(t)^H$:

$$\boldsymbol{R}(t)^H = \boldsymbol{C}_{yy}(t)\,\boldsymbol{\Theta}(t) \tag{14}$$

where $\boldsymbol{C}_{yy}(t)$ is the positive definite factor and $\boldsymbol{\Theta}(t)$ is the orthonormal factor. Now suppose that $\boldsymbol{W}(t-1)$ *approximately* spans the dominant subspace of $\boldsymbol{C}_{xx}(t)$. Then equations (13) and (14) become approximations:

$$\boldsymbol{W}(t) \simeq \boldsymbol{W}(t-1)\,\boldsymbol{\Theta}(t) \tag{15}$$

$$\boldsymbol{R}(t)^H \simeq \boldsymbol{C}_{yy}(t)\,\boldsymbol{\Theta}(t) \tag{16}$$

where the $r \times r$ matrix $\boldsymbol{\Theta}(t)$ is *nearly* orthonormal.

Compared to equation (15), the classical projection approximation [22] is equivalent to $\boldsymbol{W}(t) \simeq \boldsymbol{W}(t-1)$ at each time step[4]. The validity of this approximation additionally requires that $\boldsymbol{\Theta}(t)$ is close to the $r \times r$ identity matrix (herein denoted $\boldsymbol{I}_r$). In this case, equation (16) shows that $\boldsymbol{R}(t)^H$ must be nearly positive definite[5]. Consequently, the choice of the square root $\boldsymbol{R}(t)^H$ of $\boldsymbol{\Phi}(t)$ is restricted (*e.g.* $\boldsymbol{R}(t)$ can no longer be upper triangular, as it was in [6]).

The NP3 implementation of the power method [8] is based on this approximation, but this algorithm relies on a matrix $\boldsymbol{R}(t)$ which deviates from the positive definite structure constraint. Therefore, the classical projection approximation does not stand, and this subspace tracker is not guaranteed to converge.

Concurrently, the algorithms presented in section V do not have to face this limitation, since they rely on the less restrictive approximation (15). Also note that (15) is the best approximation of $\boldsymbol{W}(t)$ in terms of mean square error, since the solution to the minimization problem

$$\arg \min_{\boldsymbol{\Theta} \in \mathbb{C}^{r \times r}} \|\boldsymbol{W}(t) - \boldsymbol{W}(t-1)\,\boldsymbol{\Theta}\|_F^2$$

is $\boldsymbol{\Theta}(t) = \boldsymbol{W}(t-1)^H \boldsymbol{W}(t)$ (where $\boldsymbol{W}(t-1)$ is supposed to be orthonormal).

V. Approximated Power Iteration

The complexity of the power iteration method can be reduced by introducing approximation (15) at time $t-1$ in step (9). Then the $n \times r$ matrix $\boldsymbol{C}_{xy}(t)$ can be computed recursively, as shown in section V-A, and factorization (10) can be updated, as shown in section V-C. This fast update requires the introduction of a $r \times r$ auxiliary matrix $\boldsymbol{Z}(t)$, introduced in section V-B.

[4]In fact, the projection approximation in [22] is defined as $\boldsymbol{W}(t')^H \boldsymbol{x}(t) \approx \boldsymbol{W}(t-1)^H \boldsymbol{x}(t) \triangleq \boldsymbol{y}(t)\ \forall t' \geq t$. It was shown in [8, pp. 301] that this approximation is equivalent to $\boldsymbol{W}(t) \simeq \boldsymbol{W}(t-1)$ at each time step.

[5]Conversely, if $\boldsymbol{R}(t)^H$ is chosen close to the only positive definite square root of $\boldsymbol{\Phi}(t)$, the approximate polar decomposition (16) shows that $\boldsymbol{\Theta}(t) \simeq \boldsymbol{I}_r$, so that equation (15) yields $\boldsymbol{W}(t) \simeq \boldsymbol{W}(t-1)$.

A. Recursion for the matrix $\boldsymbol{C}_{xy}(t)$

It is shown in this section that the $n \times r$ matrix $\boldsymbol{C}_{xy}(t)$ can be updated in the same way as the $n \times n$ matrix $\boldsymbol{C}_{xx}(t)$ in equation (4):

$$\boldsymbol{C}_{xy}(t) = \beta\,\boldsymbol{C}_{xy}(t-1)\,\boldsymbol{\Theta}(t-1) + \underline{\boldsymbol{x}}(t)\,\underline{J}\,\underline{\widehat{\boldsymbol{y}}}(t)^H. \tag{17}$$

In the exponential window case, equation (17) involves a rank-one update ($\underline{\boldsymbol{x}}(t)$ and $\underline{\widehat{\boldsymbol{y}}}(t)$ are vectors and $\underline{J}$ is a scalar), whereas in the truncated window case it involves a rank-two update ($\underline{\boldsymbol{x}}(t)$ and $\underline{\widehat{\boldsymbol{y}}}(t)$ are two-column matrices and $\underline{J}$ is a 2×2 matrix).

1) Truncated window: first, equation (2) can be written

$$\boldsymbol{C}_{xx}(t) = \boldsymbol{X}(t)\,\boldsymbol{D}\,\boldsymbol{X}(t)^H \tag{18}$$

where $\boldsymbol{X}(t) \triangleq [\boldsymbol{x}(t-l+1), \boldsymbol{x}(t-l+2), \ldots, \boldsymbol{x}(t)]$ is the $n \times l$ data matrix and $\boldsymbol{D}$ is the $l \times l$ diagonal matrix $\mathrm{diag}(\beta^{l-1}, \beta^{l-2}, \ldots, \beta, 1)$.

Substituting equation (18) into equation (9) yields

$$\boldsymbol{C}_{xy}(t) = \boldsymbol{X}(t)\,\boldsymbol{D}\,\boldsymbol{Y}(t)^H \tag{19}$$

where $\boldsymbol{Y}(t) \triangleq \boldsymbol{W}(t-1)^H \boldsymbol{X}(t)$ is the $r \times l$ compressed data matrix. Now let us show recursions for matrices $\boldsymbol{X}(t)$ and $\boldsymbol{Y}(t)$. The first one is straightforward:

$$\left[\ \boldsymbol{x}(t-l)\ \middle|\ \boldsymbol{X}(t)\ \right] = \left[\ \boldsymbol{X}(t-1)\ \middle|\ \boldsymbol{x}(t)\ \right]. \tag{20}$$

Then left multiplying equation (20) by $\boldsymbol{W}(t-1)^H$ yields

$$\left[\ \boldsymbol{v}(t-l)\ \middle|\ \boldsymbol{Y}(t)\ \right] = \left[\ \boldsymbol{W}(t-1)^H \boldsymbol{X}(t-1)\ \middle|\ \boldsymbol{y}(t)\ \right] \tag{21}$$

where $\boldsymbol{y}(t)$, defined in equation (11), and

$$\boldsymbol{v}(t-l) \triangleq \boldsymbol{W}(t-1)^H \boldsymbol{x}(t-l) \tag{22}$$

are r-dimensional compressed data vectors. Applying approximation (15) at time $t-1$ to equation (21) yields the recursion $\left[\ \boldsymbol{v}(t-l)\ \middle|\ \boldsymbol{Y}(t)\ \right] \simeq \left[\ \widehat{\boldsymbol{V}}(t-1)\ \middle|\ \boldsymbol{y}(t)\ \right]$, where $\widehat{\boldsymbol{V}}(t-1)$ is the $r \times l$ compressed data matrix

$$\widehat{\boldsymbol{V}}(t-1) \triangleq \boldsymbol{\Theta}(t-1)^H \boldsymbol{Y}(t-1). \tag{23}$$

From now on, the exact definition of $\boldsymbol{Y}(t)$ is therefore replaced by

$$\left[\ \widehat{\boldsymbol{v}}(t-l)\ \middle|\ \boldsymbol{Y}(t)\ \right] \triangleq \left[\ \widehat{\boldsymbol{V}}(t-1)\ \middle|\ \boldsymbol{y}(t)\ \right] \tag{24}$$

where the r-dimensional vector $\widehat{\boldsymbol{v}}(t-l)$, defined by the first column in the left side of equation (24), is an approximation of the vector $\boldsymbol{v}(t-l)$. Equations (19), (20), (23) and (24) finally yield

$$\begin{aligned} \boldsymbol{C}_{xy}(t) = \ & \beta\,\boldsymbol{C}_{xy}(t-1)\,\boldsymbol{\Theta}(t-1) \\ & + \boldsymbol{x}(t)\,\boldsymbol{y}(t)^H - \beta^l\,\boldsymbol{x}(t-l)\,\widehat{\boldsymbol{v}}(t-l)^H \end{aligned} \tag{25}$$

This recursion can be seen as a particular case of equation (17), where $\underline{J}$ and $\underline{\boldsymbol{x}}(t)$ are defined in equations (7) and (8) and the $r \times p$ (with $p = 2$) matrix

$$\underline{\widehat{\boldsymbol{y}}}(t) \triangleq \left[\ \boldsymbol{y}(t)\ \middle|\ \widehat{\boldsymbol{v}}(t-l)\ \right] \tag{26}$$

is an approximation of

$$\underline{\boldsymbol{y}}(t) \triangleq \boldsymbol{W}(t-1)^H \underline{\boldsymbol{x}}(t) = \left[\ \boldsymbol{y}(t)\ \middle|\ \boldsymbol{v}(t-l)\ \right]. \tag{27}$$

2) Exponential window: substituting equation (1) into equation (9) yields

$$\boldsymbol{C}_{xy}(t) = \beta\, \boldsymbol{C}_{xx}(t-1)\boldsymbol{W}(t-1) + \boldsymbol{x}(t)\, \boldsymbol{y}(t)^H. \quad (28)$$

Applying the projection approximation (15) at time $t-1$, equation (28) can be replaced by the following recursion:

$$\boldsymbol{C}_{xy}(t) = \beta\, \boldsymbol{C}_{xy}(t-1)\, \boldsymbol{\Theta}(t-1) + \boldsymbol{x}(t)\, \boldsymbol{y}(t)^H. \quad (29)$$

This recursion can be seen as a particular case of equation (17), where $\underline{J}$ and $\underline{\boldsymbol{x}}(t)$ are defined in equations (5) and (6) and the $r \times p$ (with $p = 1$) matrix $\widehat{\underline{\boldsymbol{y}}}(t) \triangleq \boldsymbol{y}(t)$ is now equal to the vector $\underline{\boldsymbol{y}}(t) \triangleq \boldsymbol{W}(t-1)^H \underline{\boldsymbol{x}}(t) = \boldsymbol{y}(t)$.

B. Recursion for the matrix $\boldsymbol{Z}(t)$

Now, we aim at updating factorization (10) by means of equation (17). This calculation requires the introduction of an auxiliary matrix, denoted $\boldsymbol{Z}(t)$. Let $\boldsymbol{S}(t-1) \triangleq (\boldsymbol{R}(t-1)\, \boldsymbol{\Theta}(t-1))^H$ and suppose that the $r \times r$ matrix $\boldsymbol{S}(t-1)$ is non-singular. Then let

$$\boldsymbol{Z}(t-1) \triangleq \boldsymbol{S}(t-1)^{-1}. \quad (30)$$

Proposition 5.1: The $r \times r$ matrix

$$\boldsymbol{S}(t) \triangleq (\boldsymbol{R}(t)\, \boldsymbol{\Theta}(t))^H \quad (31)$$

is non-singular if and only if the $p \times p$ matrix $\beta \underline{J}^{-1} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)$ is non-singular, where

$$\underline{\boldsymbol{h}}(t) \triangleq \boldsymbol{Z}(t-1)\, \widehat{\underline{\boldsymbol{y}}}(t). \quad (32)$$

has dimension $r \times p$. In this case, the $r \times r$ matrix

$$\boldsymbol{Z}(t) \triangleq \boldsymbol{S}(t)^{-1} \quad (33)$$

satisfies the recursion

$$\boldsymbol{Z}(t) = \frac{1}{\beta} \boldsymbol{\Theta}(t)^H \left(\boldsymbol{I}_r - \underline{\boldsymbol{g}}(t)\, \underline{\boldsymbol{y}}(t)^H\right) \boldsymbol{Z}(t-1)\, \boldsymbol{\Theta}(t)^{-H} \quad (34)$$

where $\underline{\boldsymbol{g}}(t)$ is the $r \times p$ matrix

$$\underline{\boldsymbol{g}}(t) \triangleq \underline{\boldsymbol{h}}(t)\, \left(\beta \underline{J}^{-1} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)\right)^{-1}. \quad (35)$$

Proof:

Substituting equation (10) into equation (17) and left multiplying by $\boldsymbol{W}(t-1)^H$ leads to

$$\boldsymbol{\Theta}(t)\, \boldsymbol{R}(t) = \beta\, \boldsymbol{S}(t-1)^H + \underline{\boldsymbol{y}}(t)\, \underline{J}\, \widehat{\underline{\boldsymbol{y}}}(t)^H. \quad (36)$$

Next, the following matrix inversion lemma [33, pp. 18-19] will be applied to invert the right member of this equality. The interest of this approach is that the $r \times r$ matrix inversion problem is converted into a smaller $p \times p$ matrix inversion (with $p = 1$ or 2).

Lemma 5.2: Let $\boldsymbol{A}$ be a $r \times r$ non-singular complex matrix. Consider the $r \times r$ matrix $\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{P}\, \boldsymbol{J}\, \boldsymbol{Q}$, where $\boldsymbol{P}$, $\boldsymbol{J}$ and $\boldsymbol{Q}$ have dimensions $r \times m$, $m \times m$ and $m \times r$, and $\boldsymbol{J}$ is supposed to be non-singular. Then $\boldsymbol{B}$ is non-singular if and only if $\boldsymbol{J}^{-1} + \boldsymbol{Q}\, \boldsymbol{A}^{-1}\, \boldsymbol{P}$ is non-singular, and in this case

$$\boldsymbol{B}^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\, \boldsymbol{P} \left(\boldsymbol{J}^{-1} + \boldsymbol{Q}\, \boldsymbol{A}^{-1}\, \boldsymbol{P}\right)^{-1} \boldsymbol{Q}\, \boldsymbol{A}^{-1}.$$

Lemma 5.2 applied to equation (36) shows that the $r \times r$ matrix $\boldsymbol{\Theta}(t)\, \boldsymbol{R}(t)$ is non-singular if and only if the $p \times p$ matrix $\beta \underline{J}^{-1} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)$ is non-singular (which provides a fast way of detecting the singularity of $\boldsymbol{R}(t)$ or $\boldsymbol{\Theta}(t)$). In the non-singular case, lemma 5.2 leads to the equation

$$(\boldsymbol{\Theta}(t)\, \boldsymbol{R}(t))^{-1} = \frac{1}{\beta} \boldsymbol{Z}(t-1)^H \left(\boldsymbol{I}_r - \underline{\boldsymbol{y}}(t)\, \underline{\boldsymbol{g}}(t)^H\right).$$

Finally, left multiplying the complex conjugate transpose of this last equation by $\boldsymbol{\Theta}(t)^H$ and right multiplying it by $\boldsymbol{\Theta}(t)^{-H}$ yields recursion (34). ■

C. Recursion for the matrix $\boldsymbol{W}(t)$

Next, proposition 5.3 introduces a fast update for the subspace weighting matrix.

Proposition 5.3: If the $p \times p$ matrix $\beta \underline{J}^{-1} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)$ is non-singular, $\boldsymbol{W}(t)$ satisfies the recursion

$$\boldsymbol{W}(t) = \left(\boldsymbol{W}(t-1) + \underline{\boldsymbol{e}}(t)\, \underline{\boldsymbol{g}}(t)^H\right) \boldsymbol{\Theta}(t) \quad (37)$$

where $\underline{\boldsymbol{e}}(t)$ is the $n \times p$ matrix

$$\underline{\boldsymbol{e}}(t) \triangleq \underline{\boldsymbol{x}}(t) - \boldsymbol{W}(t-1)\, \underline{\boldsymbol{y}}(t). \quad (38)$$

Proof:

Substituting equation (10) into equation (17) and right multiplying by $\boldsymbol{\Theta}(t)$ shows that $\boldsymbol{W}(t)$ satisfies the recursion

$$\boldsymbol{W}(t)\boldsymbol{S}(t)^H = \left(\beta \boldsymbol{W}(t-1)\boldsymbol{S}(t-1)^H + \underline{\boldsymbol{x}}(t)\, \underline{J}\, \widehat{\underline{\boldsymbol{y}}}(t)^H\right) \boldsymbol{\Theta}(t)$$

Substituting equations (36) and (38) into the above equation yields

$$\boldsymbol{W}(t)\boldsymbol{S}(t)^H = \boldsymbol{W}(t-1)\boldsymbol{\Theta}(t)\boldsymbol{S}(t)^H + \underline{\boldsymbol{e}}(t)\, \underline{J}\, \widehat{\underline{\boldsymbol{y}}}(t)^H \boldsymbol{\Theta}(t). \quad (39)$$

However, left multiplying (36) by $\underline{\boldsymbol{g}}(t)^H$ and replacing $\underline{\boldsymbol{g}}(t)$ by its definition in equation (35) leads to

$$\begin{aligned} \underline{\boldsymbol{g}}(t)^H \boldsymbol{\Theta}(t) \boldsymbol{R}(t) = &\left(\beta \underline{J}^{-1} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)\right)^{-H} \\ &\left(\left(\beta \boldsymbol{S}(t-1) \underline{\boldsymbol{h}}(t)\right)^H + \left(\underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)\right)^H \underline{J} \widehat{\underline{\boldsymbol{y}}}(t)^H\right). \end{aligned} \quad (40)$$

Then equations (32) and (30) show that

$$(\beta \boldsymbol{S}(t-1) \underline{\boldsymbol{h}}(t))^H = \beta\, \widehat{\underline{\boldsymbol{y}}}(t)^H = \beta \underline{J}^{-1}\, \underline{J}\, \widehat{\underline{\boldsymbol{y}}}(t)^H. \quad (41)$$

Substituting equation (41) into equation (40) yields

$$\underline{\boldsymbol{g}}(t)^H\, \boldsymbol{\Theta}(t)\, \boldsymbol{R}(t) = \underline{J}\, \widehat{\underline{\boldsymbol{y}}}(t)^H. \quad (42)$$

Finally, substituting equation (42) into equation (39) and right multiplying by $\boldsymbol{S}(t)^{-H} = \boldsymbol{Z}(t)^H$ yields equation (37). ■

Note that if $\beta\, \underline{J}^{-1} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)$ is singular, $\boldsymbol{Z}(t)$ and $\boldsymbol{W}(t)$ can no longer be updated with equations (34) and (37). In practice, we never encountered this rank deficiency case in our numerical simulations[6].

[6] A solution consists in computing $\boldsymbol{W}(t)$ and $\boldsymbol{R}(t)$ by means of a SVD or a QR factorization of $\boldsymbol{C}_{xy}(t)$. Then $\boldsymbol{\Theta}(t) = \boldsymbol{W}(t-1)^H \boldsymbol{W}(t)$ can be deduced. Note that the whole processing requires $O(nr^2)$ operations; this technique must be used while $\boldsymbol{R}(t)$ or $\boldsymbol{\Theta}(t)$ remains singular. When both $\boldsymbol{R}(t)$ and $\boldsymbol{\Theta}(t)$ become non-singular again, then $\boldsymbol{Z}(t)$ can be computed, and the algorithm can switch back to the fully adaptive processing.

TABLE I
EXPONENTIAL WINDOW API ALGORITHM

Initialization :[8]

$$\boldsymbol{W}(0) = \begin{bmatrix} \boldsymbol{I}_r \\ \boldsymbol{0}_{(n-r)\times r} \end{bmatrix},\ \boldsymbol{Z}(0) = \boldsymbol{I}_r$$

For each time step do

Input vector : $\boldsymbol{x}(t)$

		Cost
PAST main section		
$\boldsymbol{y}(t) = \boldsymbol{W}(t-1)^H \boldsymbol{x}(t)$	(11)	nr
$\boldsymbol{h}(t) = \boldsymbol{Z}(t-1)\,\boldsymbol{y}(t)$	(32)	r^2
$\boldsymbol{g}(t) = \frac{\boldsymbol{h}(t)}{\beta + \boldsymbol{y}(t)^H \boldsymbol{h}(t)}$	(35)	$2r$
API main section		
$\boldsymbol{e}(t) = \boldsymbol{x}(t) - \boldsymbol{W}(t-1)\,\boldsymbol{y}(t)$	(38)	nr
$\boldsymbol{\Theta}(t) = \left(\boldsymbol{I}_r + \|\boldsymbol{e}(t)\|^2 \boldsymbol{g}(t)\,\boldsymbol{g}(t)^H\right)^{-\frac{1}{2}}$	(43)	$n + O(r^3)$
$\boldsymbol{Z}(t) = \begin{matrix} \frac{1}{\beta}\,\boldsymbol{\Theta}(t)^H \left(\boldsymbol{I}_r - \boldsymbol{g}(t)\,\boldsymbol{y}(t)^H\right) \\ \boldsymbol{Z}(t-1)\,\boldsymbol{\Theta}(t)^{-H} \end{matrix}$	(34)	$O(r^3)$
$\boldsymbol{W}(t) = \left(\boldsymbol{W}(t-1) + \boldsymbol{e}(t)\,\boldsymbol{g}(t)^H\right)\boldsymbol{\Theta}(t)$	(37)	$nr^2 + nr$

Since $\boldsymbol{W}(t-1)$ is orthonormal, $\underline{\boldsymbol{e}}(t)$ is orthogonal to $\boldsymbol{W}(t-1)$. Moreover, the orthonormality of $\boldsymbol{W}(t)$, associated to equation (37), yields

$$\boldsymbol{\Theta}(t)\,\boldsymbol{\Theta}(t)^H = \left(\boldsymbol{I}_r + \underline{\boldsymbol{g}}(t)\,\left(\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)\right)\,\underline{\boldsymbol{g}}(t)^H\right)^{-1}. \quad (43)$$

Therefore, $\boldsymbol{\Theta}(t)$ is an inverse square root of the $r \times r$ positive definite matrix $\boldsymbol{I}_r + \underline{\boldsymbol{g}}(t)\,\left(\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)\right)\,\underline{\boldsymbol{g}}(t)^H$. The choice of this inverse square root does not affect the subspace tracking performance[7].

The pseudo-code of the exponential window API algorithm is presented in table I, and that of the truncated window API algorithm (TW-API) is presented in table II. It can be noted that the first section of API is exactly the same as that of the PAST subspace tracker [22]; it requires only $nr + r^2 + O(r)$ operations per time step, while the rest of the algorithm has a $nr^2 + o(nr^2)$ computational complexity. In the same way, the first section of TW-API is similar to that of the sliding window version of PAST [29]; it requires only $2nr + 2r^2 + O(r)$, while the rest of the algorithm has a $(n+l)r^2 + o(nr^2)$ computational complexity. Note that the implementations of API and TW-API presented in tables I and II are of limited interest, since a number of faster subspace trackers have already been proposed in the literature, which have a $O(nr)$ complexity (among which [3], [22], [23], [25], [29], [34] are illustrated in section VIII). A faster implementation of API and TW-API is proposed in section VI.

[7]Let $\boldsymbol{\Theta}^P(t)$ be the only positive definite inverse square root. Then $\boldsymbol{\Theta}(t)$ can be written in the form

$$\boldsymbol{\Theta}(t) = \boldsymbol{\Theta}^P(t)\,\boldsymbol{U}(t) \quad (44)$$

where $\boldsymbol{U}(t)$ is a $r \times r$ orthonormal matrix. Substituting equation (44) into equation (37) yields

$$\boldsymbol{W}(t) = \left\{\left(\boldsymbol{W}(t-1) + \boldsymbol{e}(t)\,\boldsymbol{g}(t)^H\right)\boldsymbol{\Theta}^P(t)\right\}\boldsymbol{U}(t).$$

It can be readily seen in this last equation that $\boldsymbol{U}(t)$ does not affect the subspace spanned by $\boldsymbol{W}(t)$; it only affects the particular orthonormal basis $\boldsymbol{W}(t)$ of this subspace. Consequently, the choice of a particular inverse square root $\boldsymbol{\Theta}(t)$ has no impact on the subspace tracking performance.

[8]The initial values $\boldsymbol{W}(0)$ and $\boldsymbol{Z}(0)$ have to be chosen suitably:

- $\boldsymbol{W}(0)$ should be a $n \times r$ orthonormal matrix,
- $\boldsymbol{Z}(0)$ should be a $r \times r$ positive definite matrix.

Both matrices can be calculated from an initial block of data or from arbitrary initial data. The simplest way, however, is to set $\boldsymbol{W}(0) = \begin{bmatrix} \boldsymbol{I}_r \\ \boldsymbol{0}_{(n-r)\times r} \end{bmatrix}$ and $\boldsymbol{Z}(0) = \boldsymbol{I}_r$. The choice of these initial values affects the transient behavior but not the steady state performance of the algorithm.

TABLE II
TRUNCATED WINDOW API (TW-API) ALGORITHM

Initialization :

$$\boldsymbol{W}(0) = \begin{bmatrix} \boldsymbol{I}_r \\ \boldsymbol{0}_{(n-r)\times r} \end{bmatrix},\ \boldsymbol{Z}(0) = \boldsymbol{I}_r,\ \boldsymbol{X}(0) = \boldsymbol{0}_{n\times l},\ \widehat{\boldsymbol{V}}(0) = \boldsymbol{0}_{r\times l}$$

For each time step do

Input vector : $\boldsymbol{x}(t)$

		Cost
Section similar to SW − PAST		
$\begin{bmatrix} \boldsymbol{x}(t-l) \mid \boldsymbol{X}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}(t-1) \mid \boldsymbol{x}(t) \end{bmatrix}$	(20)	
$\boldsymbol{y}(t) = \boldsymbol{W}(t-1)^H \boldsymbol{x}(t)$	(11)	nr
$\begin{bmatrix} \widehat{\boldsymbol{v}}(t-l) \mid \widehat{\boldsymbol{Y}}(t) \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{V}}(t-1) \mid \boldsymbol{y}(t) \end{bmatrix}$	(24)	
$\boldsymbol{v}(t-l) = \boldsymbol{W}(t-1)^H \boldsymbol{x}(t-l)$	(22)	nr
$\underline{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{x}(t) \mid \boldsymbol{x}(t-l) \end{bmatrix}$	(8)	
$\underline{\widehat{\boldsymbol{y}}}(t) = \begin{bmatrix} \boldsymbol{y}(t) \mid \widehat{\boldsymbol{v}}(t-l) \end{bmatrix}$	(26)	
$\underline{\boldsymbol{y}}(t) = \begin{bmatrix} \boldsymbol{y}(t) \mid \boldsymbol{v}(t-l) \end{bmatrix}$	(27)	
$\underline{\boldsymbol{h}}(t) = \boldsymbol{Z}(t-1)\,\underline{\widehat{\boldsymbol{y}}}(t)$	(32)	$2r^2$
$\underline{\boldsymbol{g}}(t) = \underline{\boldsymbol{h}}(t)\,\left(\beta\,\underline{J}^{-1} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)\right)^{-1}$	(35)	$8r$
TW − API main section		
$\underline{\boldsymbol{e}}(t) = \underline{\boldsymbol{x}}(t) - \boldsymbol{W}(t-1)\,\underline{\boldsymbol{y}}(t)$	(38)	$2nr$
$\boldsymbol{\Theta}(t) = \left(\boldsymbol{I}_r + \underline{\boldsymbol{g}}(t)\,\left(\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)\right)\,\underline{\boldsymbol{g}}(t)^H\right)^{-\frac{1}{2}}$	(43)	$4n + O(r^3)$
$\boldsymbol{Z}(t) = \begin{matrix} \frac{1}{\beta}\,\boldsymbol{\Theta}(t)^H \left(\boldsymbol{I}_r - \underline{\boldsymbol{g}}(t)\,\underline{\boldsymbol{y}}(t)^H\right) \\ \boldsymbol{Z}(t-1)\,\boldsymbol{\Theta}(t)^{-H} \end{matrix}$	(34)	$O(r^3)$
$\boldsymbol{W}(t) = \left(\boldsymbol{W}(t-1) + \underline{\boldsymbol{e}}(t)\,\underline{\boldsymbol{g}}(t)^H\right)\boldsymbol{\Theta}(t)$	(37)	$nr^2 + 2nr$
$\widehat{\boldsymbol{V}}(t) = \boldsymbol{\Theta}(t)^H \widehat{\boldsymbol{Y}}(t)$	(23)	lr^2

VI. FAST API METHOD

In this section, a fast implementation of the API method is proposed, based on a particular choice of the matrix $\boldsymbol{\Theta}(t)$. It is supposed that $\beta\,\underline{J}^{-1} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)$ is non-singular, so that $\boldsymbol{\Theta}(t)$ is also non-singular. Below, the $p \times p$ identity matrix is denoted $\underline{I}_p$.

A. A particular solution to equation (43)

Let $\underline{\varepsilon}(t)$ be a square root of the $p \times p$ matrix $\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)$:

$$\underline{\varepsilon}(t)\,\underline{\varepsilon}(t)^H = \left(\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)\right). \quad (45)$$

Substituting equation (45) into equation (43) and applying the matrix inversion lemma shows that[9]

$$\boldsymbol{\Theta}(t)\,\boldsymbol{\Theta}(t)^H = \boldsymbol{I}_r - \underline{\boldsymbol{g}}(t)\,\underline{\varepsilon}(t)\,\underline{\rho}(t)^{-1}\underline{\varepsilon}(t)^H \underline{\boldsymbol{g}}(t)^H \quad (46)$$

where $\underline{\rho}(t)$ is the $p \times p$ positive definite matrix

$$\underline{\rho}(t) = \underline{I}_p + \underline{\varepsilon}(t)^H\,\left(\underline{\boldsymbol{g}}(t)^H \underline{\boldsymbol{g}}(t)\right)\underline{\varepsilon}(t). \quad (47)$$

Considering equation (46), we are looking for a special solution of the form

$$\boldsymbol{\Theta}(t) = \boldsymbol{I}_r - \underline{\boldsymbol{g}}(t)\,\underline{\varepsilon}(t)\,\underline{\sigma}(t)^{-1}\underline{\varepsilon}(t)^H \underline{\boldsymbol{g}}(t)^H \quad (48)$$

[9]Lemma 5.2 is applied with $\boldsymbol{A} = \boldsymbol{I}_r$, $\boldsymbol{P} = \underline{\boldsymbol{g}}(t)\,\underline{\varepsilon}(t)$, $\boldsymbol{J} = \underline{I}_p$ and $\boldsymbol{Q} = \underline{\varepsilon}(t)^H \underline{\boldsymbol{g}}(t)^H$. In particular, the non-singularity of $\boldsymbol{\Theta}(t)$ is equivalent to the non-singularity of $\underline{\rho}(t)$.

where $\underline{\sigma}(t)$ is a $p \times p$ non-singular matrix. The interest of this approach is that the $r \times r$ matrix square rooting problem in equation (46) is converted into a smaller $p \times p$ matrix square rooting. Indeed, substituting equation (48) into equation (46) yields a sufficient condition :

$$\underline{\sigma}(t)^{-1} + \underline{\sigma}(t)^{-H} + \underline{\sigma}(t)^{-1} \left(\underline{I}_p - \underline{\rho}(t)\right) \underline{\sigma}(t)^{-H} = \underline{\rho}(t)^{-1}.$$

Left multiplying the two members of this last equation by $\underline{\sigma}(t)$ and right multiplying them by $\underline{\sigma}(t)^H$ yields the equation[10]

$$\left(\underline{\sigma}(t) - \underline{\rho}(t)\right) \underline{\rho}(t)^{-1} \left(\underline{\sigma}(t) - \underline{\rho}(t)\right)^H = \underline{I}_p$$

whose solution is

$$\underline{\sigma}(t) = \underline{\rho}(t) + \underline{\rho}(t)^{\frac{1}{2}^H}. \tag{49}$$

Even if other choices would be possible, from now on we suppose that the square root of $\underline{\rho}(t)$ which is involved in the above equation is the only positive definite square root. This condition guarantees that $\underline{\sigma}(t)$ is positive definite, so that $\boldsymbol{\Theta}(t)$ is hermitian[11]. Then define the $p \times p$ positive definite matrix

$$\underline{\tau}(t) = \underline{\varepsilon}(t)\, \sigma(t)^{-1} \underline{\varepsilon}(t)^H. \tag{50}$$

Substituting equation (50) into equation (48) yields

$$\boldsymbol{\Theta}(t) = \boldsymbol{I}_r - \underline{\boldsymbol{g}}(t)\, \underline{\tau}(t)\, \underline{\boldsymbol{g}}(t)^H. \tag{51}$$

B. Fast implementation of the particular solution

Based on the low-rank matrix update of $\boldsymbol{\Theta}(t)$ in equation (51), it is shown below that the matrices $\boldsymbol{Z}(t)$, $\boldsymbol{W}(t)$ and $\widehat{\boldsymbol{V}}(t)$ can also be efficiently updated. Consider the $p \times p$ matrix

$$\underline{\eta}(t) = \underline{I}_p - \left(\underline{\boldsymbol{g}}(t)^H \underline{\boldsymbol{g}}(t)\right) \underline{\tau}(t). \tag{52}$$

Since $\boldsymbol{\Theta}(t)$ is non-singular, the matrix inversion lemma shows that $\underline{\eta}(t)$ is also non-singular[12]. Then substituting equation (51) into equation (34) yields

$$\boldsymbol{Z}(t) = \frac{1}{\beta} \left(\boldsymbol{Z}(t-1) - \underline{\boldsymbol{g}}(t)\, \underline{\boldsymbol{h}}'(t)^H + \underline{\boldsymbol{\epsilon}}(t)\, \underline{\boldsymbol{g}}(t)^H\right) \tag{53}$$

where the $r \times p$ matrices $\underline{\boldsymbol{h}}'(t)$ and $\underline{\boldsymbol{\epsilon}}(t)$ are defined by

$$\begin{aligned}
\underline{\boldsymbol{y}}'(t) &= \underline{\boldsymbol{y}}(t)\, \underline{\eta}(t) + \underline{\boldsymbol{g}}(t)\, \underline{\tau}(t) && (54)\\
\underline{\boldsymbol{h}}'(t) &= \boldsymbol{Z}(t-1)^H \underline{\boldsymbol{y}}'(t) && (55)\\
\underline{\boldsymbol{\epsilon}}(t) &= \left(\boldsymbol{Z}(t-1)\underline{\boldsymbol{g}}(t) - \underline{\boldsymbol{g}}(t) \left(\underline{\boldsymbol{h}}'(t)^H \underline{\boldsymbol{g}}(t)\right)\right) \\
&\quad \left(\underline{\tau}(t)\, \underline{\eta}(t)^{-1}\right)^H && (56)
\end{aligned}$$

Then substituting equation (51) into equation (37) yields

$$\boldsymbol{W}(t) = \boldsymbol{W}(t-1) + \underline{\boldsymbol{e}}'(t)\, \underline{\boldsymbol{g}}(t)^H \tag{57}$$

where $\underline{\boldsymbol{e}}'(t)$ is the $n \times p$ matrix

$$\underline{\boldsymbol{e}}'(t) = \underline{\boldsymbol{e}}(t)\, \underline{\eta}(t) - \boldsymbol{W}(t-1)\, \underline{\boldsymbol{g}}(t)\, \underline{\tau}(t). \tag{58}$$

However, substituting equations (38) and (54) into equation (58) yields

$$\underline{\boldsymbol{e}}'(t) = \underline{\boldsymbol{x}}(t)\, \underline{\eta}(t) - \boldsymbol{W}(t-1)\, \underline{\boldsymbol{y}}'(t). \tag{59}$$

Finally, substituting equation (51) into equation (23) yields

$$\widehat{\boldsymbol{V}}(t) = \boldsymbol{Y}(t) - \underline{\boldsymbol{g}}(t) \left(\underline{\boldsymbol{g}}(t)\, \underline{\tau}(t)\right)^H \boldsymbol{Y}(t). \tag{60}$$

The pseudo-code of the exponential window fast API algorithm (FAPI) is presented in table III, and that of the truncated window fast API algorithm (TW-FAPI) is presented in table IV. The overall computational cost of FAPI is $n(3r+2) + 5r^2 + O(r)$ flops per iteration[13] (whereas the complexities of PAST [22] and OPAST [25] are respectively $3nr + 2r^2 + O(r)$ and $n(4r+1) + 2r^2 + O(r)$). The overall computational cost of TW-FAPI is $n(6r+8) + 4lr + O(r^2)$ flops per iteration[14] (whereas the complexities of SW-PAST and SW-OPAST [29] are respectively $5nr + 4r^2 + O(r)$ and $n(15r+28) + 12r^2 + O(r)$). Note that the presence of a $4lr$ term in the complexity of TW-FAPI may make this algorithm more computationally demanding in applications for which l is much larger than n. However, in the context of frequency estimation, it has been proved that optimal Cramer-Rao bounds were obtained for $\frac{1}{2}n \leq l \leq 2n$ [35], and in section VIII-A, TW-FAPI is tested with $l = \frac{3}{2}n$.

TABLE III
EXPONENTIAL WINDOW FAST API (FAPI) ALGORITHM

Initialization (*cf.* table I)

For each time step do

Input vector : $\boldsymbol{x}(t)$

PAST main section (*cf.* table I)

FAPI main section :		**Cost**
$\varepsilon^2(t) = \|\boldsymbol{x}(t)\|^2 - \|\boldsymbol{y}(t)\|^2$	(45)	$n + r$
$\tau(t) = \frac{\varepsilon^2(t)}{1+\varepsilon^2(t)\|\boldsymbol{g}(t)\|^2 + \sqrt{1+\varepsilon^2(t)\|\boldsymbol{g}(t)\|^2}}$	(50)	r
$\eta(t) = 1 - \tau(t)\, \|\boldsymbol{g}(t)\|^2$	(52)	1
$\boldsymbol{y}'(t) = \eta(t)\, \boldsymbol{y}(t) + \tau(t)\, \boldsymbol{g}(t)$	(54)	$2r$
$\boldsymbol{h}'(t) = \boldsymbol{Z}(t-1)^H \boldsymbol{y}'(t)$	(55)	r^2
$\boldsymbol{\epsilon}(t) = \frac{\tau(t)}{\eta(t)} \left(\boldsymbol{Z}(t-1)\boldsymbol{g}(t) - \left(\boldsymbol{h}'(t)^H \boldsymbol{g}(t)\right) \boldsymbol{g}(t)\right)$	(56)	$r^2 + 3r$
$\boldsymbol{Z}(t) = \frac{1}{\beta} \left(\boldsymbol{Z}(t-1) - \boldsymbol{g}(t)\, \boldsymbol{h}'(t)^H + \boldsymbol{\epsilon}(t)\, \boldsymbol{g}(t)^H\right)$	(53)	$2r^2$
$\boldsymbol{e}'(t) = \eta(t)\, \boldsymbol{x}(t) - \boldsymbol{W}(t-1)\, \boldsymbol{y}'(t)$	(59)	$nr + n$
$\boldsymbol{W}(t) = \boldsymbol{W}(t-1) + \boldsymbol{e}'(t)\, \boldsymbol{g}(t)^H$	(57)	nr

VII. LINK WITH THE PAST AND OPAST ALGORITHMS

In this section, it is shown that the classical exponential window PAST algorithm can be seen as a first order approximation of the FAPI algorithm. Indeed, the error $\boldsymbol{e}(t)$ is the component of $\boldsymbol{x}(t)$ that does not belong to the signal subspace spanned by $\boldsymbol{W}(t-1)$. Thus, if this subspace slowly varies upon time, and if the Signal to Noise Ratio (SNR) is high, $\boldsymbol{e}(t) \simeq \boldsymbol{0}$. If the second order term $\|\boldsymbol{e}(t)\|^2$ is disregarded in

[10] Remember that $\underline{\rho}(t)$ is an hermitian matrix.

[11] More precisely, $\boldsymbol{\Theta}(t)$ is positive definite. Indeed, equation (49) shows that $\underline{\sigma}(t)$ and $\underline{\rho}(t)$ are simultaneously diagonalizable, and the eigenvalues of $\underline{\sigma}(t)$ are strictly greater than those of $\underline{\rho}(t)$. Therefore, $\underline{\rho}(t)^{-1} - \underline{\sigma}(t)^{-1}$ is a positive definite matrix. Then subtracting equation (46) from equation (48) shows that $\boldsymbol{\Theta}(t)$ is positive definite.

[12] Lemma 5.2 is applied to equation (51), with $\boldsymbol{A} = \boldsymbol{I}_r$, $\boldsymbol{P} = \underline{\boldsymbol{g}}(t)\, \underline{\tau}(t)$, $\boldsymbol{J} = \underline{I}_p$ and $\boldsymbol{Q} = \underline{\boldsymbol{g}}(t)^H$.

[13] Note that this implementation of FAPI is faster than that proposed in [26], whose global cost was $n(4r+2) + 5r^2 + O(r)$.

[14] This implementation of TW-FAPI is also faster than that proposed in [31], whose global cost was $n(8r+8) + 4lr + O(r^2)$.

TABLE IV
TRUNCATED WINDOW FAST API (TW-FAPI) ALGORITHM

Initialization (*cf.* table II)

For each time step do

Section similar to SW − PAST (*cf.* table II)

TW − FAPI main section		**Cost**
$\underline{\varepsilon}(t) = \left(\underline{\boldsymbol{x}}(t)^H \underline{\boldsymbol{x}}(t) - \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{y}}(t)\right)^{\frac{1}{2}}$	(45)	$4n + 4r$
$\underline{\rho}(t) = \underline{I}_p + \underline{\varepsilon}(t)^H \left(\underline{\boldsymbol{g}}(t)^H \underline{\boldsymbol{g}}(t)\right) \underline{\varepsilon}(t)$	(47)	$4r$
$\underline{\tau}(t) = \underline{\varepsilon}(t) \left(\underline{\rho}(t) + \underline{\rho}(t)^{\frac{1}{2}^H}\right)^{-1} \underline{\varepsilon}(t)^H$	(50)	$O(1)$
$\underline{\eta}(t) = \underline{I}_p - \left(\underline{\boldsymbol{g}}(t)^H \underline{\boldsymbol{g}}(t)\right) \underline{\tau}(t)$	(52)	$O(1)$
$\underline{\boldsymbol{y}}'(t) = \underline{\boldsymbol{y}}(t)\, \underline{\eta}(t) + \underline{\boldsymbol{g}}(t)\, \underline{\tau}(t)$	(54)	$8r$
$\underline{\boldsymbol{h}}'(t) = \boldsymbol{Z}(t-1)^H \underline{\boldsymbol{y}}'(t)$	(55)	$2r^2$
$\underline{\boldsymbol{\epsilon}}(t) = \left(\boldsymbol{Z}(t-1)\underline{\boldsymbol{g}}(t) - \underline{\boldsymbol{g}}(t)\left(\underline{\boldsymbol{h}}'(t)^H \underline{\boldsymbol{g}}(t)\right)\right) \left(\underline{\tau}(t)\, \underline{\eta}(t)^{-1}\right)^H$	(56)	$2r^2 + 12r$
$\boldsymbol{Z}(t) = \frac{1}{\beta}\left(\boldsymbol{Z}(t-1) - \underline{\boldsymbol{g}}(t)\, \underline{\boldsymbol{h}}'(t)^H + \underline{\boldsymbol{\epsilon}}(t)\, \underline{\boldsymbol{g}}(t)^H\right)$	(53)	$4r^2$
$\underline{\boldsymbol{e}}'(t) = \underline{\boldsymbol{x}}(t)\, \underline{\eta}(t) - \boldsymbol{W}(t-1)\, \underline{\boldsymbol{y}}'(t)$	(59)	$2nr + 4n$
$\boldsymbol{W}(t) = \boldsymbol{W}(t-1) + \underline{\boldsymbol{e}}'(t)\, \underline{\boldsymbol{g}}(t)^H$	(57)	$2nr$
$\widehat{\boldsymbol{V}}(t) = \boldsymbol{Y}(t) - \underline{\boldsymbol{g}}(t) \left(\underline{\boldsymbol{g}}(t)\, \underline{\tau}(t)\right)^H \boldsymbol{Y}(t)$	(60)	$4lr$

table III, $\tau(t) = 0$, $\eta(t) = 1$ and $\boldsymbol{\Theta}(t)$ becomes the $r \times r$ identity matrix. Then equations (57) and (53) become

$$\boldsymbol{W}(t) = \boldsymbol{W}(t-1) + \boldsymbol{e}(t)\, \boldsymbol{g}(t)^H \tag{61}$$

$$\boldsymbol{Z}(t) = \frac{1}{\beta}\left(\boldsymbol{Z}(t-1) - \boldsymbol{g}(t)\, \boldsymbol{h}(t)^H\right) \tag{62}$$

(in particular, it can be recursively shown that $\boldsymbol{Z}(t)$ is always hermitian). Consequently, this first order approximation of the fast API method is an exact implementation of the classical PAST subspace tracker [22], which only provides a *nearly* orthonormal subspace weighting matrix. In other respects, a thorough examination of the OPAST algorithm presented in [25] shows that $\boldsymbol{W}(t)$ is updated as in equation (57) (which guarantees the orthonormality, contrary to equation (61)). However, $\boldsymbol{Z}(t)$ is updated as in equation (62). Consequently, OPAST can be seen as an intermediary between PAST and FAPI.

VIII. SIMULATION RESULTS

In this section, the performance of the subspace estimation is analyzed in the context of frequency estimation, in terms of the maximum principal angle between the true dominant subspace of the correlation matrix $\boldsymbol{C}_{xx}(t)$ (obtained via an exact eigenvalue decomposition), and the estimated dominant subspace of the same correlation matrix (obtained with the subspace tracker). This error criterion was initially proposed by P. Comon and G.H. Golub as a measure of the distance between equidimensional subspaces [24, pp. 603-604]). In section VIII-A, the FAPI and TW-FAPI algorithms are compared to other existing subspace trackers. In section VIII-B, the behavior of the API method regarding the SNR and the parameters n and r is investigated.

A. Comparison of FAPI and TW-FAPI with other existing subspace trackers

In this section, the test signal is a sum of $r = 4$ complex sinusoidal sources plus a complex white gaussian noise (the SNR is 5.7 dB). The frequencies of the sinusoids vary according to a jump scenario originally proposed by P. Strobach in the context of Direction Of Arrival estimation [36]: their values abruptly change at different time instants, between which they remain constant. Their variations are represented on Figure 1-a. This signal is processed in section VIII-A.1 by means of an exponential window whose forgetting factor is $\beta \approx 0.99$, and in section VIII-A.2 by means of a sliding window of length $l = 120$. This parameters were chosen so that the effective window length is the same in both cases, *i.e.* $\beta = \frac{1}{1-1/l}$. Section VIII-A.3 focuses on the orthonormality of the subspace weighting matrix. The complexities of the various subspace trackers illustrated in this section are given in table V.

Algorithm	Complexity (flops)	Window	Figure
FAPI	$n(3r+2) + 5r^2 + O(r)$	exponential	Fig. 1
PAST	$3nr + 2r^2 + O(r)$		
NIC	$4nr + 2r^2 + O(r)$		
OPAST	$n(4r+1) + 2r^2 + O(r)$		
Karasalo	$nr^2 + n(3r+2) + O(r^3)$	exponential	Fig. 2
FST	$n(6r+2) + 12r^2 + O(r)$		
Householder PAST	$n(4r+1) + 2r^2 + O(r)$		
Loraf2	$nr^2 + n(3r+2) + O(r^3)$		
SP1	$4nr^2 + n(4r+2) + O(r^3)$		
TW-FAPI	$n(6r+8) + 4lr + O(r^2)$	sliding	Fig. 3
SW-PAST	$5nr + 4r^2 + O(r)$		
SW-NIC	$6nr + 4r^2 + O(r)$		
SW-OPAST	$n(15r+28) + 12r^2 + O(r)$		

TABLE V
COMPARISON OF THE COMPLEXITIES

1) Exponential window case: figure 1-b shows the maximum principal angle error trajectory $\theta_{\mathrm{FAPI}}(t)$, obtained with the FAPI method with parameters $n = 80$ and $\beta \approx 0.99$. Then this result is compared to that obtained with the PAST subspace tracker: figure 1-c shows the ratio in dB of the trajectories obtained with FAPI and PAST, *i.e.*

$$20 \log_{10}\left(\frac{\theta_{\mathrm{FAPI}}(t)}{\theta_{\mathrm{PAST}}(t)}\right).$$

At initialization, it can be noticed that FAPI converges faster than PAST. Moreover, PAST does not provide an orthonormal subspace weighting matrix. Figure 1-d shows the ratio in dB of the trajectories obtained with FAPI and the NIC subspace tracker[15], which is a robust generalization of PAST [23]. It can be seen that the subspace estimation error is always smaller with FAPI. As PAST, NIC does not guarantee the orthonormality of the subspace weighting matrix. Figure 1-e shows the ratio of the trajectories obtained with FAPI and OPAST. The two algorithms reach the same performance, except at initialization, where FAPI converges faster. In fact, the difference is much more distinct with the sliding window versions of these algorithms (see section VIII-A.2).

[15]The learning step η is equal to 0.7.

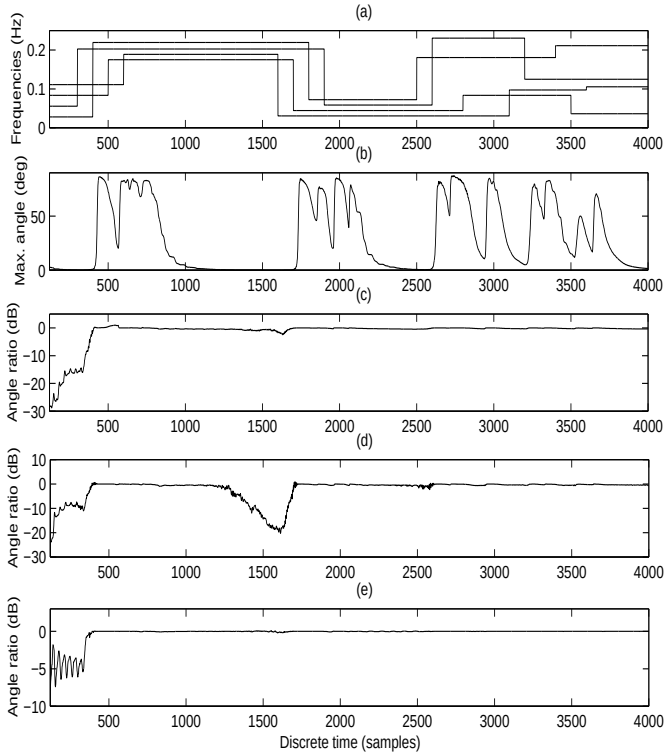

Fig. 1. Subspace tracking based on an exponential window
(a) Normalized frequencies of the sinusoids
(b) Maximum principal angle trajectory obtained with FAPI
(c) Ratio of the trajectories obtained with FAPI and PAST
(d) Ratio of the trajectories obtained with FAPI and NIC
(e) Ratio of the trajectories obtained with FAPI and OPAST

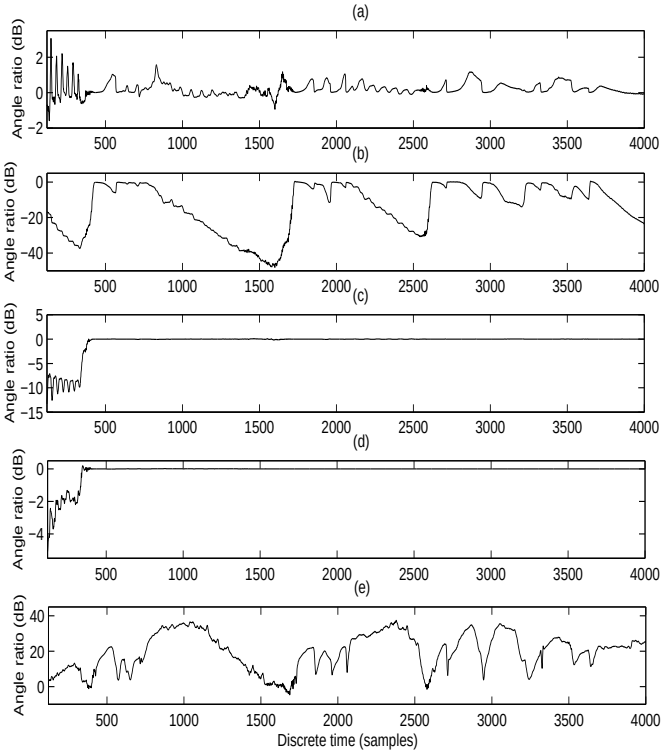

Fig. 2. Subspace tracking based on an exponential window
(a) Ratio of the trajectories obtained with FAPI and Karasalo
(b) Ratio of the trajectories obtained with FAPI and FST
(c) Ratio of the trajectories obtained with FAPI and Householder PAST
(d) Ratio of the trajectories obtained with FAPI and Loraf2
(e) Ratio of the trajectories obtained with FAPI and SP1

In figure 2, the FAPI algorithm is compared to five other well-known subspace trackers :

- I. Karasalo's algorithm [2],
- the Fast Subspace Tracking (FST) algorithm [3],
- the novel PAST algorithm employing Householder transformations, herein called Householder PAST [34],
- the Low-Rank Adaptive Filter (Loraf2) algorithm [7],
- and the Subspace Projection (SP1) algorithm [37].

Figure 2-a shows that the behaviors of FAPI and Karasalo's algorithm are very similar. However the dominant cost of the latter is nr^2 (see table V). Figure 2-b shows that FAPI converges to the signal subspace much more precisely than FST. Moreover, FST is more computationally demanding than FAPI. Figure 2-c shows that FAPI and Householder PAST reach the same performance, except at initialization, where FAPI converges faster. Figure 2-d shows that the same remark can be made about FAPI and Loraf2. Besides, the dominant complexity of Loraf2 is nr^2.

Among the various subspace trackers that we have tested, SP1 is the only one which really outperformed FAPI (see figure 2-e). However, table V shows that SP1 is the most computationally demanding algorithm. In other respects, it is only suitable for time series data analysis, and was only designed for exponential windows.

2) Sliding window case: figure 3-a shows the maximum principal angle error trajectory $\theta_{\mathrm{TW-FAPI}}(t)$, obtained with the TW-FAPI method with parameters $\beta = 1$ (which turns the truncated window into a sliding window), $n = 80$ and $l = 120$. It can be noticed that this algorithm has a fast convergence rate after each frequency jump. This result can be compared to that of figure 1-b, obtained with the exponential window FAPI method, for which the response to frequency jumps is slower, because of the nature of the window which tends to smooth the signal variations. Figure 3-b shows the ratio in dB of the trajectories obtained with TW-FAPI and the sliding window version of PAST, herein called SW-PAST [22], [29]. It can be seen that TW-FAPI converges faster than SW-PAST at initialization. Note that as PAST, SW-PAST does not provide an orthonormal subspace weighting matrix. Figure 3-c shows the ratio in dB of the trajectories obtained with TW-FAPI and a sliding window version of the NIC algorithm, herein called SW-NIC[16]. Finally, figure 3-d shows the ratio in dB of the trajectories obtained with TW-FAPI and the sliding window OPAST algorithm [29]. It can be noticed that the maximum principal angle error trajectory obtained with TW-FAPI is about 20 dB lower than those obtained with SW-NIC and SW-OPAST in regions where the frequencies are constant.

3) Orthonormality error: the orthonormality of the subspace weighting matrix $\boldsymbol{W}(t)$ can be measured by means of the following error criterion:

$$20\log_{10}\left(\|\boldsymbol{W}(t)^H\boldsymbol{W}(t) - \boldsymbol{I}_r\|_F\right).$$

[16]SW-NIC is also implemented with $\eta = 0.7$.

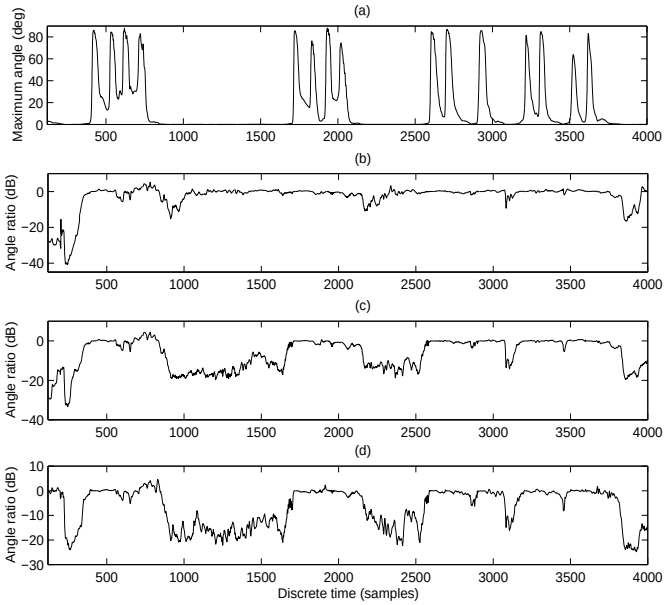

Fig. 3. Subspace tracking based on a sliding window
(a) Maximum principal angle trajectory obtained with TW-FAPI
(b) Ratio of the trajectories obtained with TW-FAPI and SW-PAST
(c) Ratio of the trajectories obtained with TW-FAPI and SW-NIC
(d) Ratio of the trajectories obtained with TW-FAPI and SW-OPAST

Algorithms	Orthonormality error
FAPI, TW-FAPI, OPAST, Householder PAST	about -300 dB
Karasalo, FST, Loraf2	about -280 dB
SP1, SW-OPAST	about -240 dB
PAST, NIC	about -25 dB
SW-PAST, SW-NIC	about -5 dB

TABLE VI
MAXIMUM ORTHONORMALITY ERROR

Table VI shows the maximum orthonormality error reached by the above mentioned algorithms while tracking the test signal variations. We observed that FAPI, TW-FAPI, OPAST and Householder PAST outperformed all the other algorithms, whereas PAST, NIC, and their sliding window versions do not guarantee the orthonormality of the subspace weighting matrix.

B. Behavior of the API method regarding the SNR and the parameters n and r

In this section, the test signal is still a sum of $r = 4$ complex sinusoidal sources plus a complex white gaussian noise. However, the frequencies of the sinusoids are constant, equal to the initial values given in figure 1-a.

1) Influence of the SNR: in this section, the effect of the SNR onto the subspace estimation is investigated. To this end, the noise part of the test signal was synthesized so that the SNR varies linearly from +30 dB to -30 dB (see figure 4-a).

Figure 4-b shows the maximum principal angle error trajectory obtained with the FAPI method with parameters $n = 80$ and $\beta \approx 0.99$. It can be seen that the performance of the subspace estimation collapses beyond $n \simeq 2600$. Figure 4-a shows that from this time instant the SNR is lower than -10 dB. Figure 4-c shows the maximum principal angle error trajectory obtained with the TW-FAPI method with parameters $\beta = 1$, $n = 80$ and $l = 120$. Again, the performance of the subspace estimation collapses beyond $n \simeq 2600$. Although they are not illustrated here, we observed that the performance of all the above mentioned subspace trackers similarly collapse beyond the same SNR limit (-10 dB).

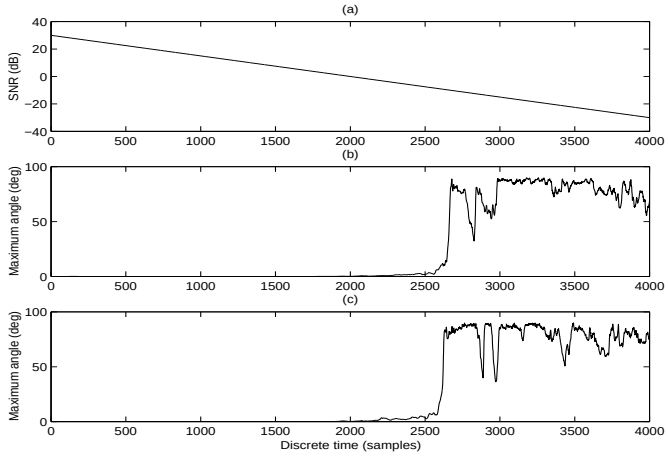

Fig. 4. Influence of the Signal to Noise Ratio
(a) Signal to Noise Ratio in dB
(b) Maximum principal angle trajectory obtained with FAPI
(c) Maximum principal angle trajectory obtained with TW-FAPI

2) Influence of the ratio n/r: in this section, we focus on the influence of the ratio n/r onto the subspace estimation. The SNR is constant, equal to 5.7 dB.

Figure 5-a shows the mean of $\theta_{\mathrm{FAPI}}(t)$, as a function of the ratio n/r, for all $n \in \{r+1, \ldots, 80\}$ (with $\beta \approx 0.99$). It can be seen that the subspace estimation becomes reliable as soon as $n/r \geq 7$. Figure 5-b shows the mean of $\theta_{\mathrm{TW-FAPI}}(t)$, as a function of the ratio n/r, for all $n \in \{r+1, \ldots, 80\}$ (with $\beta = 1$ and $l = 120$). Again, it can be seen that the subspace estimation becomes reliable as soon as $n/r \geq 7$. Although they are not illustrated here, we observed that the same remark is valid for all the above mentioned subspace trackers.

3) Tracking a subspace of wrong dimension: since the dimension r of the signal subspace is unknown in many applications, we investigate in this section the performance of the FAPI and TW-FAPI algorithms when applied with a wrong subspace dimension r. The SNR is constant, equal to 5.7 dB. The performance of the subspace estimation is analyzed in terms of the maximum principal angle between the true 4-dimensional signal subspace and the estimated r-dimensional subspace.

Figure 5-c shows the mean of $\theta_{\mathrm{FAPI}}(t)$, as a function of r, for all $r \in \{1, \ldots, 20\}$ (with parameters $\beta \approx 0.99$ and $n = 80$). Similarly, figure 5-d shows the mean of $\theta_{\mathrm{TW-FAPI}}(t)$, as a function of r, for all $r \in \{1, \ldots, 20\}$ (with parameters $l = 120$ and $n = 80$). It can be seen that the subspace estimation is reliable in all cases:

- if $r = 4$, the maximum principal angle is very low (as expected),
- if $r < 4$, the maximum principal angle remains low, which means that the estimated lower-dimensional subspace is nearly included in the true signal subspace,

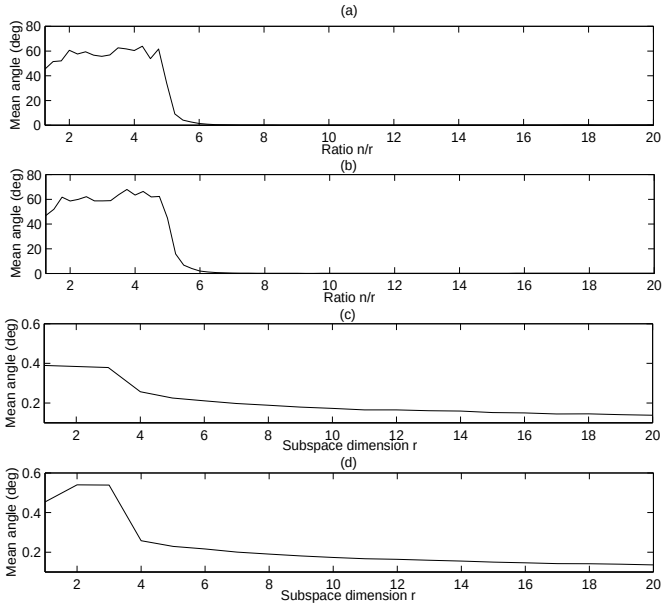

Fig. 5. Influence of the parameters n and r
(a) Average max. angle obtained with FAPI as a function of n/r
(b) Average max. angle obtained with TW-FAPI as a function of n/r
(c) Average max. angle obtained with FAPI as a function of r
(d) Average max. angle obtained with TW-FAPI as a function of r

- is $r > 4$, the maximum principal angle is even lower than in the case $r = 4$, which means that the true signal subspace is nearly included in the estimated upper-dimensional subspace. Moreover, it can be noticed that the maximum principal angle decreases as the dimension of the estimated subspace increases.

We can conclude that FAPI and TW-FAPI are robust to erroneous subspace dimension r.

IX. Conclusions

In this paper, several implementations of the API algorithm for subspace tracking were presented, based either on exponential windows or on truncated windows. These algorithms reach a linear complexity and guarantee the orthonormality of the subspace weighting matrix at each time step. In the context of frequency estimation, the method proves able to track abrupt frequency variations robustly, and outperforms many subspace trackers, both in terms of subspace estimation and computational complexity. Finally, these subspace tracking algorithms can be considered as the starting point of a real-time frequency tracker, whose full implementation can involve our adaptive version of the ESPRIT algorithm [38].

References

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

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

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

[4] M. Moonen, P. V. Dooren, and J. Vandewalle, "An SVD updating algorithm for subspace tracking," *SIAM J. Matrix Ana. Appl.*, vol. 13, no. 4, pp. 1015–1038, 1992.

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

[6] P. Strobach, "Low-rank adaptive filters," *IEEE Trans. Signal Processing*, vol. 44, no. 12, pp. 2932–2947, Dec. 1996.

[7] ——, "Bi-iteration SVD subspace tracking algorithms," *IEEE Trans. Signal Processing*, vol. 45, no. 5, pp. 1222–1240, May 1997.

[8] 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*, Oct. 1999.

[9] C. H. Bischof and G. M. Shroff, "On updating signal subspaces," *IEEE Trans. Signal Processing*, vol. 40, pp. 96–105, 1992.

[10] G. W. Stewart, "An updating algorithm for subspace tracking," *IEEE Trans. Signal Processing*, vol. 40, pp. 1535–1541, June 1992.

[11] G. Xu, H. Zha, G. H. Golub, and T. Kailath, "Fast algorithms for updating signal subspaces," *IEEE Trans. Circuits Syst.*, vol. 41, no. 8, pp. 537–549, Aug. 1994.

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

[13] T. Chonavel, B. Champagne, and C. Riou, "Fast adaptive eigenvalue decomposition: a maximum likelihood approach," *Signal processing*, vol. 83, no. 2, pp. 307–324, Feb. 2003.

[14] C. S. MacInnes, "Fast, accurate subspace tracking using operator restriction analysis," in *Proc. of IEEE Int. Conf. on Acoustic, Speech and Signal Processing*, 1998, pp. 1357–1360.

[15] B. Champagne, "SVD-updating via constrained perturbations with application to subspace tracking," *Signals, Systems and Computers*, vol. 2, pp. 1379–1385, 1996.

[16] E. Oja, "Neural networks, principal components and subspaces," *Int. journal of neural systems*, vol. 1, no. 1, pp. 61–68, 1989.

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

[18] T. Chen and S. Amari, "Unified stabilization approach to principal and minor components extraction algorithms," *Neural Networks*, vol. 14, no. 10, pp. 1377–1387, 2001.

[19] S. Y. Kung, K. I. Diamantaras, and J. S. Taur, "Adaptive principal component extraction (apex) and applications," *IEEE Trans. Signal Processing*, vol. 43, no. 1, pp. 1202–1217, Jan. 1995.

[20] G. Mathew and V. U. Reddy, "Adaptive estimation of eigensubspace," *IEEE Trans. Signal Processing*, vol. 43, no. 2, pp. 401–411, Feb. 1995.

[21] Z. Fu and E. M. Dowling, "Conjugate gradient eigenstructure tracking for adaptive spectral estimation," *IEEE Trans. Signal Processing*, vol. 43, no. 5, pp. 1151–1160, May 1995.

[22] B. Yang, "Projection Approximation Subspace Tracking," *IEEE Trans. Signal Processing*, vol. 44, no. 1, pp. 95–107, Jan. 1995.

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

[24] G. H. Golub and C. F. V. Loan, *Matrix computations*, 3rd ed. Baltimore and London: The Johns Hopkins University Press, 1996.

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

[26] R. Badeau, G. Richard, and B. David, "Approximated power iterations for fast subspace tracking," in *Proc. of 7th Int. Symp. on Signal Proc. and its Applications*, vol. 2, Paris, France, July 2003, pp. 583–586.

[27] P. Strobach, "Square hankel SVD subspace tracking algorithms," *Signal Processing*, vol. 57, no. 1, pp. 1–18, Feb. 1997.

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

[29] R. Badeau, K. Abed-Meraim, G. Richard, and B. David, "Sliding Window Orthonormal PAST Algorithm," in *Proc. of IEEE Int. Conf. on Acoustic, Speech and Signal Processing*, vol. 5, Apr. 2003, pp. 261–264.

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

[31] ——, "Suivi d'espace dominant par la méthode des puissances itérées," in *Proc. of 19ème colloque GRETSI sur le traitement du signal et des images*, vol. 1, Sept. 2003, pp. 137–140.

[32] R. O. Schmidt, "A signal subspace approach to multiple emitter location and spectral estimation," Ph.D. dissertation, Stanford University, Nov. 1981.

[33] R. A. Horn and C. R. Johnson, *Matrix analysis*. Cambridge: Cambridge University Press, 1985.

[34] S. C. Douglas, "Numerically-robust adaptive subspace tracking using Householder transformations," in *Proc. of IEEE Sensor Array and Multichannel Signal Proc. Workshop*, 2000, pp. 499 –503.

[35] Y. Hua and T. K. Sarkar, "Matrix pencil method for estimating parameters of exponentially damped/undamped sinusoids in noise," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 38, no. 5, pp. 814–824, May 1990.

[36] P. Strobach, "Fast recursive subspace adaptive ESPRIT algorithms," *IEEE Trans. Signal Processing*, vol. 46, no. 9, pp. 2413–2430, Sept. 1998.

[37] C. E. Davila, "Efficient, high performance, subspace tracking for time-domain data," *IEEE Trans. Signal Processing*, vol. 48, no. 12, pp. 3307–3315, Dec. 2000.

[38] R. Badeau, G. Richard, and B. David, "Adaptive ESPRIT algorithm based on the PAST subspace tracker," in *Proc. of IEEE ICASSP'03*, vol. 6, Apr. 2003, pp. 229–232.