# APPROXIMATED POWER ITERATIONS FOR FAST SUBSPACE TRACKING

*Roland Badeau, Gaël Richard, Bertrand David and Karim Abed-Meraim*

Ecole Nationale Supérieure des Télécommunications
46 rue Barrault, 75634 Paris Cedex 13 France
rbadeau, grichard, bedavid, abed@tsi.enst.fr

## ABSTRACT

This paper introduces a fast implementation of the power iterations method for subspace tracking, based on an approximation less restrictive than the well known *projection approximation*. This algorithm guarantees the orthonormality of the estimated subspace weighting matrix at each iteration, and satisfies a global and exponential convergence property. Moreover, it outperforms many subspace trackers related to the power method, such as PAST, NIC, NP3 and OPAST, while keeping the same computational complexity.

## 1. INTRODUCTION

Subspace tracking has been widely investigated in the fields of adaptive filtering, source localization or parameter estimation. One of the various approaches proposed in the literature consists in the iterative optimization of a specific cost function involving the estimated covariance matrix of the data, in conjunction with a projection approximation hypothesis (see *e.g.* the PAST [1] and NIC [2] methods).

In [3], it has been shown that these subspace trackers are closely linked to the classical power iterations method [4]. A fast implementation of this method was proposed in [3], but numerical simulations showed that this algorithm (referred to as NP3) does not converge in many situations. Concurrently, the Orthonormal PAST (OPAST) algorithm [5, 6] is an other fast implementation of the power method which outperforms both PAST and NP3.

In this paper, we propose a new subspace tracker based on the power method and on a new mild projection approximation, which has the same computational complexity as the above mentioned algorithms, but reaches better performances than NIC and OPAST.

The paper is organized as follows. In section 2 the classical power iterations method is summarized. The projection approximation is discussed in section 3. Then our approximated power iterations (API) method is introduced in section 4, and a fast implementation of this algorithm is proposed in section 5. Section 6 shows that both PAST and OPAST can be viewed as approximations of the fast API algorithm. In section 7, the performance of this method is compared to that of NIC and OPAST. Finally, the main conclusions of this paper are summarized in section 8.

## 2. THE BASIC POWER ITERATIONS METHOD

Let $\{\boldsymbol{x}(t)\}_{t\geq 0}$ be a sequence of $n$ dimensional data vectors, whose estimated covariance matrix is recursively updated according to the following scheme:

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

where $\beta > 0$ is the forgetting factor. If $\boldsymbol{W}(t-1)$ is a $n \times r$ orthonormal matrix (with $r < n$) spanning the principal subspace of $\boldsymbol{C}_{xx}(t-1)$, the compressed data vector is defined as

$$\boldsymbol{y}(t) = \boldsymbol{W}(t-1)^H \boldsymbol{x}(t). \tag{2}$$

To track the subspace weighting matrix $\boldsymbol{W}(t)$, the power iterations method [3] consists of a data compression step (3) plus an orthonormalization step (4) at each iteration:

$$\boldsymbol{C}_{xy}(t) = \boldsymbol{C}_{xx}(t)\,\boldsymbol{W}(t-1) \tag{3}$$

$$\boldsymbol{W}(t)\,\boldsymbol{R}(t) = \boldsymbol{C}_{xy}(t) \tag{4}$$

where $\boldsymbol{C}_{xy}(t)$ can be seen as a $n \times r$ covariance matrix, and $\boldsymbol{R}(t)^H$ is a square root of the $r \times r$ hermitian matrix $\boldsymbol{\Phi}(t) \triangleq \boldsymbol{C}_{xy}(t)^H \boldsymbol{C}_{xy}(t)$, which means that $\boldsymbol{R}(t)^H\,\boldsymbol{R}(t) = \boldsymbol{\Phi}(t)$. Note that $\boldsymbol{R}(t)^H$ can be any square root of $\boldsymbol{\Phi}(t)$ (for example, $\boldsymbol{R}(t)$ can be triangular [7], or hermitian [3]).

If $\boldsymbol{C}_{xx}(t)$ remains constant and if its first $r$ eigenvalues are strictly larger than the $(n-r)^{\text{th}}$ other ones, the power iterations method converges globally and exponentially to the principal subspace [3][4, pp. 410-411]. Note that the multiplication in the first step involves $O(n^2 r)$ operations, and the orthonormalization step requires $O(nr^2)$ operations. Because of its high computational cost, this algorithm is not suitable for real-time processing.

## 3. THE PROJECTION APPROXIMATION

The classical projection approximation [1] has been used in conjunction with various subspace trackers in order to reduce their complexity. It assumes that $\boldsymbol{W}(t) \simeq \boldsymbol{W}(t-1)$ at each time step. In the case of the power method, this approximation yields $\boldsymbol{W}(t-1)^H \boldsymbol{C}_{xx}(t) \boldsymbol{W}(t-1) \simeq \boldsymbol{R}(t)$. Consequently, $\boldsymbol{R}(t)^H$ can no longer be any square root of $\boldsymbol{\Phi}(t)$, since it must be close to a non-negative hermitian matrix[1].

The NP3 implementation of the power method [3] was based on this approximation, but this algorithm relies on a matrix $\boldsymbol{R}(t)$ which deviates from the non-negative hermitian structure constraint. Therefore, the projection approximation does not stand, and this subspace tracker may not converge. To solve this problem, consider the less restrictive approximation

$$\boldsymbol{\Pi}_{\boldsymbol{W}(t)} \simeq \boldsymbol{\Pi}_{\boldsymbol{W}(t-1)} \tag{5}$$

where $\boldsymbol{\Pi}_{\boldsymbol{W}(t)} \triangleq \boldsymbol{W}(t)\,\boldsymbol{W}(t)^H$ is the projection onto the range space of $\boldsymbol{W}(t)$. Equation (5) yields the *new projection approximation* (NPA):

$$\boldsymbol{W}(t) \simeq \boldsymbol{W}(t-1)\,\boldsymbol{\Theta}(t) \tag{6}$$

[1]Conversely, it can be shown that if $\boldsymbol{R}(t)^H$ is the non-negative hermitian square root of $\boldsymbol{\Phi}(t)$, then $\boldsymbol{W}(t) \simeq \boldsymbol{W}(t-1)$.

where $\boldsymbol{\Theta}(t) \triangleq \boldsymbol{W}(t-1)^H \boldsymbol{W}(t)$ is not necessarily close to the identity. Since $\boldsymbol{W}(t-1)^H \boldsymbol{C}_{xx}(t) \boldsymbol{W}(t-1) = \boldsymbol{\Theta}(t) \boldsymbol{R}(t)$, the non-negative hermitian structure constraint is now related to the product $\boldsymbol{\Theta}(t)\,\boldsymbol{R}(t)$, and $\boldsymbol{R}(t)^H$ can be any square root of $\boldsymbol{\Phi}(t)$, like in the exact power method. Based on the NPA, the API method will be introduced in the next section. It will be shown that this algorithm appromatively satisfies the structure constraint on the matrix product $\boldsymbol{\Theta}(t)\,\boldsymbol{R}(t)$.

## 4. APPROXIMATED POWER ITERATIONS

Substituting equation (1) into equation (3) yields

$$\boldsymbol{C}_{xy}(t) = \beta\, \boldsymbol{C}_{xx}(t-1) \boldsymbol{W}(t-1) + \boldsymbol{x}(t)\, \boldsymbol{y}(t)^H \tag{7}$$

Applying the NPA (6) at time $t-1$, equation (7) can be replaced by the following recursion:

$$\boldsymbol{C}_{xy}(t) = \beta\, \boldsymbol{C}_{xy}(t-1)\, \boldsymbol{\Theta}(t-1) + \boldsymbol{x}(t)\, \boldsymbol{y}(t)^H. \tag{8}$$

Based on this recursion, it will be shown that the factorization in equation (4) can be efficiently updated. Let $\boldsymbol{S}(t-1) \triangleq (\boldsymbol{R}(t-1)\, \boldsymbol{\Theta}(t-1))^H$ and suppose that $\boldsymbol{S}(t-1)$ is non-singular. Define the auxiliary matrix $\boldsymbol{Z}(t-1) = \boldsymbol{S}(t-1)^{-1}$, and consider the $r$ dimensional vector

$$\boldsymbol{h}(t) = \boldsymbol{Z}(t-1)\, \boldsymbol{y}(t). \tag{9}$$

Proposition 4.1 shows that $\boldsymbol{Z}(t)$ can be recursively updated.

**Proposition 4.1** *The matrix $\boldsymbol{S}(t) \triangleq (\boldsymbol{R}(t)\, \boldsymbol{\Theta}(t))^H$ is non-singular if and only if $\beta + \boldsymbol{y}(t)^H \boldsymbol{h}(t) \neq 0$, and in this case the matrix $\boldsymbol{Z}(t) \triangleq \boldsymbol{S}(t)^{-1}$ satisfies the recursion*

$$\boldsymbol{Z}(t) = \frac{1}{\beta} \boldsymbol{\Theta}(t)^H \left( \boldsymbol{I} - \boldsymbol{g}(t) \boldsymbol{y}(t)^H \right) \boldsymbol{Z}(t-1) \boldsymbol{\Theta}(t)^{-H} \tag{10}$$

*where $\boldsymbol{g}(t)$ is the $r$ dimensional vector*

$$\boldsymbol{g}(t) = \frac{\boldsymbol{h}(t)}{\beta + \boldsymbol{y}(t)^H \boldsymbol{h}(t)}. \tag{11}$$

**Proof** Substituting equation (4) into equation (8) and left multiplying by $\boldsymbol{W}(t-1)^H$ shows that

$$\boldsymbol{\Theta}(t)\, \boldsymbol{R}(t) = \beta\, \boldsymbol{S}(t-1)^H + \boldsymbol{y}(t)\, \boldsymbol{y}(t)^H. \tag{12}$$

Next, consider the following matrix inversion lemma [8, pp. 18-19]:

**Lemma 4.2** *Let $\boldsymbol{A}$ be a $r \times r$ non-singular complex matrix. Consider the $r \times r$ matrix $\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{y}\, \boldsymbol{z}^H$, where $\boldsymbol{y}$ and $\boldsymbol{z}$ are $r$ dimensional vectors. Then $\boldsymbol{B}$ is non-singular if and only if $1 + \boldsymbol{z}^H \boldsymbol{A}^{-1} \boldsymbol{y} \neq 0$, and in this case*

$$\boldsymbol{B}^{-1} = \boldsymbol{A}^{-1} - \frac{1}{1 + \boldsymbol{z}^H \boldsymbol{A}^{-1} \boldsymbol{y}} (\boldsymbol{A}^{-1} \boldsymbol{y})(\boldsymbol{A}^{-H} \boldsymbol{z})^H.$$

Lemma 4.2 applied to equation (12) shows that $\boldsymbol{\Theta}(t)\, \boldsymbol{R}(t)$ is non-singular if and only if $\beta + \boldsymbol{y}(t)^H \boldsymbol{h}(t) \neq 0$, and in this case

$$(\boldsymbol{\Theta}(t)\, \boldsymbol{R}(t))^{-1} = \frac{1}{\beta}\, \boldsymbol{Z}(t-1)^H \left( \boldsymbol{I}_r - \boldsymbol{y}(t)\, \boldsymbol{g}(t)^H \right) \tag{13}$$

In particular, detecting that $\beta + \boldsymbol{y}(t)^H \boldsymbol{h}(t) = 0$ is a fast way of detecting the singularity of $\boldsymbol{R}(t)$ or $\boldsymbol{\Theta}(t)$. In the non-singular case, equation (13) finally yields equation (10). ∎

Next, proposition 4.3 introduces a fast update for the subspace weighting matrix.

**Proposition 4.3** *In the case $\beta + \boldsymbol{y}(t)^H \boldsymbol{h}(t) \neq 0$, $\boldsymbol{W}(t)$ satisfies the recursion*

$$\boldsymbol{W}(t) = \left( \boldsymbol{W}(t-1) + \boldsymbol{e}(t)\, \boldsymbol{g}(t)^H \right) \boldsymbol{\Theta}(t) \tag{14}$$

*where $\boldsymbol{e}(t)$ is the $n$ dimensional vector*

$$\boldsymbol{e}(t) = \boldsymbol{x}(t) - \boldsymbol{W}(t-1)\, \boldsymbol{y}(t). \tag{15}$$

**Proof** Substituting equation (4) into equation (8) and right multiplying by $\boldsymbol{\Theta}(t)$ shows that $\boldsymbol{W}(t)$ satisfies the recursion

$$\boldsymbol{W}(t) \boldsymbol{S}(t)^H = \left( \beta \boldsymbol{W}(t-1) \boldsymbol{S}(t-1)^H + \boldsymbol{x}(t) \boldsymbol{y}(t)^H \right) \boldsymbol{\Theta}(t) \tag{16}$$

Substituting equations (12) and (15) into equation (16) yields

$$\boldsymbol{W}(t) \boldsymbol{S}(t)^H = \boldsymbol{W}(t-1) \boldsymbol{\Theta}(t) \boldsymbol{S}(t)^H + \boldsymbol{e}(t) \boldsymbol{y}(t)^H \boldsymbol{\Theta}(t). \tag{17}$$

Left multiplying equation (12) by $\boldsymbol{g}(t)^H$ shows that

$$\boldsymbol{y}(t)^H = \boldsymbol{g}(t)^H\, \boldsymbol{\Theta}(t)\, \boldsymbol{R}(t). \tag{18}$$

Finally, substituting equation (18) into equation (17) and right multiplying by $\boldsymbol{Z}(t)^H$ yields equation (14). ∎

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

$$\boldsymbol{\Theta}(t)\, \boldsymbol{\Theta}(t)^H = \left( \boldsymbol{I}_r + \|\boldsymbol{e}(t)\|^2 \boldsymbol{g}(t)\, \boldsymbol{g}(t)^H \right)^{-1}. \tag{19}$$

Therefore, $\boldsymbol{\Theta}(t)$ can be *any* inverse square root of the matrix $\boldsymbol{I}_r + \|\boldsymbol{e}(t)\|^2 \boldsymbol{g}(t)\, \boldsymbol{g}(t)^H$. The choice of this inverse square root will not affect the subspace tracking performance. Indeed, note that the error vector $\boldsymbol{e}(t)$ 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, $\boldsymbol{e}(t) \simeq \boldsymbol{0}$. Therefore, $\boldsymbol{\Theta}(t)$ is nearly orthonormal. Consequently, equation (10) shows that $\boldsymbol{Z}(t)$ remains nearly non-negative hermitian, and so does $\boldsymbol{\Theta}(t)\boldsymbol{R}(t)$ (see equation (12)). Finally, the API method satisfies the non-negative hermitian structure constraint mentioned in section 3.

The complete pseudo-code is presented in table 1. It can be noted that the first section of this subspace tracker is exactly the same as that of the PAST algorithm [1]. This section requires only $O(nr)$ operations, while the rest of the algorithm has a $O(nr^2)$ computational complexity.

In the particular case $\beta + \boldsymbol{y}(t)^H \boldsymbol{h}(t) = 0$, $\boldsymbol{Z}(t)$ and $\boldsymbol{W}(t)$ can no longer be updated with equations (10) and (14). 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. In practice, we never encountered the rank deficiency case in our numerical simulations.

Table 1: Approximated Power Iterations (API) method

```
Initialization :
$\boldsymbol{W}(0) = \begin{bmatrix} \boldsymbol{I}_r \\ \boldsymbol{0}_{(n-r)\times r} \end{bmatrix}, \ \boldsymbol{Z}(0) = \boldsymbol{I}_r$
for each time step do
    input vector : $\boldsymbol{x}(t)$
    PAST main section :
    $\boldsymbol{y}(t) = \boldsymbol{W}(t-1)^H \boldsymbol{x}(t)$    (2)
    $\boldsymbol{h}(t) = \boldsymbol{Z}(t-1)\,\boldsymbol{y}(t)$    (9)
    $\boldsymbol{g}(t) = \frac{\boldsymbol{h}(t)}{\beta + \boldsymbol{y}(t)^H \boldsymbol{h}(t)}$    (11)
    $\boldsymbol{e}(t) = \boldsymbol{x}(t) - \boldsymbol{W}(t-1)\,\boldsymbol{y}(t)$    (15)
    API main section :
    $\boldsymbol{\Theta}(t) = \left(\boldsymbol{I}_r + \|\boldsymbol{e}(t)\|^2 \boldsymbol{g}(t)\,\boldsymbol{g}(t)^H\right)^{-\frac{1}{2}}$    (19)
    $\boldsymbol{Z}(t) = \frac{1}{\beta}\boldsymbol{\Theta}(t)^H \left(\boldsymbol{I} - \boldsymbol{g}(t)\boldsymbol{y}(t)^H\right) \boldsymbol{Z}(t-1)\boldsymbol{\Theta}(t)^{-H}$    (10)
    $\boldsymbol{W}(t) = \left(\boldsymbol{W}(t-1) + \boldsymbol{e}(t)\,\boldsymbol{g}(t)^H\right)\boldsymbol{\Theta}(t)$    (14)
```

## 5. FAST API METHOD

In this section, a fast implementation of the API method will be proposed, based on a particular choice of the matrix $\boldsymbol{\Theta}(t)$, which reduces the overall complexity to $O(nr)$. Let

$$\tau(t) = \frac{\|\boldsymbol{e}(t)\|^2}{1 + \|\boldsymbol{e}(t)\|^2\|\boldsymbol{g}(t)\|^2 + \sqrt{1 + \|\boldsymbol{e}(t)\|^2\|\boldsymbol{g}(t)\|^2}}. \quad (20)$$

A direct calculation shows that the $r \times r$ hermitian matrix

$$\boldsymbol{\Theta}(t) \triangleq \boldsymbol{I}_r - \tau(t)\,\boldsymbol{g}(t)\,\boldsymbol{g}(t)^H \quad (21)$$

is an inverse square root of $\boldsymbol{I}_r + \|\boldsymbol{e}(t)\|^2\boldsymbol{g}(t)\,\boldsymbol{g}(t)^H$. Then substituting equation (21) into equation (10) yields

$$\boldsymbol{Z}(t) = \frac{1}{\beta}\left(\boldsymbol{Z}(t-1) - \boldsymbol{g}(t)\,\boldsymbol{h}'(t)^H + \boldsymbol{\epsilon}(t)\,\boldsymbol{g}(t)^H\right) \quad (22)$$

where $\boldsymbol{h}'(t)$ and $\boldsymbol{\epsilon}(t)$ are the $r$ dimensional vectors

$$\boldsymbol{h}'(t) = \boldsymbol{Z}(t-1)^H\left(\left(1 - \tau(t)\|\boldsymbol{g}(t)\|^2\right)\boldsymbol{y}(t) + \tau(t)\boldsymbol{g}(t)\right) \quad (23)$$

$$\boldsymbol{\epsilon}(t) = \frac{\tau(t)}{1 - \tau(t)\|\boldsymbol{g}(t)\|^2}\left(\boldsymbol{Z}(t-1)\boldsymbol{g}(t) - \left(\boldsymbol{h}'(t)^H\boldsymbol{g}(t)\right)\boldsymbol{g}(t)\right) \quad (24)$$

Finally, substituting equation (21) into equation (14) yields

$$\boldsymbol{W}(t) = \boldsymbol{W}(t-1) + \boldsymbol{e}'(t)\,\boldsymbol{g}(t)^H \quad (25)$$

where $\boldsymbol{e}'(t)$ is the $r$ dimensional vector

$$\boldsymbol{e}'(t) = \left(1 - \tau(t)\|\boldsymbol{g}(t)\|^2\right)\boldsymbol{e}(t) - \tau(t)\boldsymbol{W}(t-1)\boldsymbol{g}(t). \quad (26)$$

The fast API (FAPI) method is summarized in table 2. Its overall computational cost is $4nr + O(r^2)$ flops per iteration (whereas the complexities of PAST and OPAST are respectively $3nr + O(r^2)$ and $4nr + O(r^2)$).

Table 2: Fast API (FAPI) method

```
Initialization (cf. table 1)
for each time step do
    input vector : $\boldsymbol{x}(t)$
    PAST main section (cf. table 1)
    FAPI main section :
    $\tau(t) = \frac{\|\boldsymbol{e}(t)\|^2}{1+\|\boldsymbol{e}(t)\|^2\|\boldsymbol{g}(t)\|^2+\sqrt{1+\|\boldsymbol{e}(t)\|^2\|\boldsymbol{g}(t)\|^2}}$
    $\boldsymbol{h}'(t) = \boldsymbol{Z}(t-1)^H\left(\left(1 - \tau(t)\,\|\boldsymbol{g}(t)\|^2\right)\boldsymbol{y}(t) + \tau(t)\,\boldsymbol{g}(t)\right)$
    $\boldsymbol{\epsilon}(t) = \frac{\tau(t)}{1-\tau(t)\|\boldsymbol{g}(t)\|^2}\left(\boldsymbol{Z}(t-1)\boldsymbol{g}(t) - \left(\boldsymbol{h}'(t)^H\boldsymbol{g}(t)\right)\boldsymbol{g}(t)\right)$
    $\boldsymbol{Z}(t) = \frac{1}{\beta}\left(\boldsymbol{Z}(t-1) - \boldsymbol{g}(t)\,\boldsymbol{h}'(t)^H + \boldsymbol{\epsilon}(t)\,\boldsymbol{g}(t)^H\right)$
    $\boldsymbol{e}'(t) = \left(1 - \tau(t)\,\|\boldsymbol{g}(t)\|^2\right)\boldsymbol{e}(t) - \tau(t)\,\boldsymbol{W}(t-1)\,\boldsymbol{g}(t)$
    $\boldsymbol{W}(t) = \boldsymbol{W}(t-1) + \boldsymbol{e}'(t)\,\boldsymbol{g}(t)^H$
```

## 6. LINK WITH THE PAST AND OPAST ALGORITHMS

In this section, it will be shown that the classical PAST algorithm can be seen as a first order approximation of the fast API method. Indeed, if the second order term $\|\boldsymbol{e}(t)\|^2$ is disregarded, $\tau(t) = 0$ and $\boldsymbol{\Theta}(t)$ becomes the $r \times r$ identity matrix. Then equations (25) and (22) become

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

$$\boldsymbol{Z}(t) = \frac{1}{\beta}\left(\boldsymbol{Z}(t-1) - \boldsymbol{g}(t)\,\boldsymbol{h}(t)^H\right) \quad (27)$$

(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 [1]. In other respects, a thorough examination of the OPAST algorithm presented in [5] shows that $\boldsymbol{W}(t)$ is updated as in equation (25), but $\boldsymbol{Z}(t)$ is updated as in equation (27). Consequently, OPAST can be seen as an intermediary between PAST and FAPI.

## 7. 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 covariance matrix $\boldsymbol{C}_{xx}(t)$ (obtained via an exact eigenvalue decomposition), and the estimated dominant subspace of the same covariance 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 [4].

The test signal of Figure 1-a 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 [9]: their values abruptly change at different time instants, between which they remain constant. Their variations are represented on Figure 1-b.

Figure 1-c shows the maximum principal angle error trajectory $\theta_{\mathrm{FAPI}}(t)$, obtained with the FAPI method with parameters $n = 80$ and $\beta = 0.99$. In Figure 2-a, this result is compared to that obtained with the NIC subspace tracker[2], which can be seen as a robust generalization of PAST [2]. Figure 2-a shows the ratio in dB of the trajectories obtained with FAPI and NIC, *i.e.*

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

It can be seen that the subspace estimation error is always smaller with FAPI. Figure 2-b shows the ratio of the trajectories obtained with FAPI and OPAST. It can be seen that 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 both algorithms (the sliding window API algorithm will be presented in [10]).

Finally, the orthonormality of the subspace weighting matrix can be measured by means of the error criterion [11]

$$\eta(t) = \|\boldsymbol{W}(t)^H \boldsymbol{W}(t) - \boldsymbol{I}\|_F^2.$$

We observed on our test signal that the NIC subspace tracker reached a maximum error of $-20.5$ dB, whereas OPAST never exceeded $-295$ dB, and FAPI never exceeded $-305$ dB.

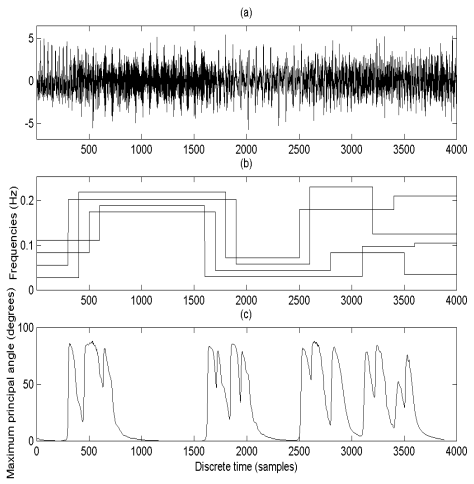

Figure 1: (a): Test signal; (b): Normalized frequencies of the sinusoids; (c): Maximum principal angle trajectory obtained with the FAPI algorithm.

## 8. CONCLUSIONS

In this paper, a fast implementation of the power iterations method for subspace tracking was presented, which guarantees the orthonormality of the estimated subspace weighting matrix at each time step. This algorithm reaches the linear complexity $O(nr)$ and satisfies a global and exponential convergence property. In the context of frequency estimation, it proved to robustly track abrupt frequency variations, and outperformed the NIC and OPAST algorithms.

[2]The learning step was equal to $0.5$.

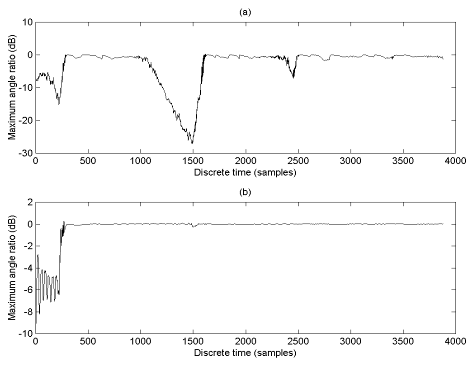

Figure 2: Ratio of the trajectories obtained with (a) FAPI and NIC; (b) FAPI and OPAST.

## 9. REFERENCES

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

[2] 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.

[3] 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.

[4] G. H. Golub and C. F. Van Loan, *Matrix computations*, The Johns Hopkins University Press, Baltimore and London, third edition, 1996.

[5] 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.

[6] 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*, Apr. 2003.

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

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

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

[10] R. Badeau, G. Richard, and B. David, "Suivi d'espace dominant par la méthode des puissances itérées," in *19ème colloque GRETSI sur le traitement du signal et des images*, Sept. 2003, to be published.

[11] 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.