Suivi d'espace dominant par la méthode des puissances itérées

Roland Badeau, Gaël Richard, Bertrand David

Suivi d'espace dominant par la méthode des puissances itérées

Roland BADEAU, Gaël RICHARD, Bertrand DAVID

École Nationale Supérieure des Télécommunications
46 rue Barrault, 75634 Paris Cedex 13 France
[roland.badeau,gael.richard,bertrand.david]@enst.fr

Résumé – Cet article introduit une version à fenêtre glissante de l'algorithme API, qui effectue le suivi de l'espace dominant d'une séquence de vecteurs. Cet algorithme est dérivé de la méthode des puissances itérées, et repose sur une approximation moins restrictive que celle connue sous le nom d'*approximation par projection*. Il garantit l'orthonormalité de la matrice générée à chaque itération, et satisfait une propriété de convergence globale et exponentielle. De plus, il atteint de meilleures performances que la plupart des algorithmes de suivi d'espace dominant voisins de la méthode des puissances itérées, tels que PAST, NIC, NP3 et OPAST, tout en ayant la même complexité algorithmique. Nos simulations numériques ont montré l'intérêt de l'utilisation d'une fenêtre glissante : l'algorithme réagit beaucoup plus rapidement à de brusques variations du signal.

Abstract – This paper introduces a sliding window version of the API subspace tracker. This algorithm is derived from the power iterations method, and relies on an approximation less restrictive than the well known *projection approximation*. It guarantees the orthonormality of the 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 having the same computational complexity. Our numerical simulations have shown the interest of using a sliding window: the algorithm offers a much faster tracking response to abrupt signal variations.

1 Introduction

Les techniques de suivi d'espace dominant ont été principalement étudiées dans les domaines du filtrage adaptatif, de la localisation de sources, et plus généralement de l'estimation de paramètres. L'une des méthodes proposées dans la littérature consiste à optimiser une fonction de coût de façon itérative. C'est le cas des algorithmes PAST [1] et NIC [2]. Dans [3], il a été démontré que ces algorithmes de suivi peuvent aussi être dérivés à partir de la méthode des puissances itérées [4]. Plusieurs implémentations de cette méthode, reposant sur des factorisations QR, ont été proposées dans [5]. Ainsi, l'algorithme Loraf2 s'avère stable et robuste, mais possède une complexité algorithmique supérieure à celle de PAST et NIC ; inversement l'algorithme Loraf3 possède la même complexité, mais au prix d'une dégradation des performances. Une autre implémentation rapide de la méthode des puissances itérées, l'algorithme NP3 reposant sur des mises à jour de rang 1, a été présenté dans [3], mais cet algorithme s'avère instable dans de nombreux cas. Une dernière implémentation, baptisée OPAST car conçue comme une version orthonormalisée de PAST, a été proposée dans [6], et fournit de meilleurs résultats que PAST, NIC et NP3. Plus récemment, nous avons présenté un nouvel algorithme basé sur la méthode des puissances itérées que nous avons baptisé API [7], et qui présente la même complexité algorithmique que PAST, NIC, NP3 et OPAST, mais fournit une estimation plus précise de l'espace dominant.

Tous les algorithmes mentionnés ci-dessus ont été conçus pour une fenêtre d'analyse à décroissance exponentielle. En effet, ce choix permet de lisser les variations du signal, et autorise à chaque instant une mise à jour peu coûteuse en temps de calcul. En comparaison, les algorithmes basés sur des fenêtres glissantes sont souvent plus coûteux, mais offrent une réponse plus rapide à de brusques variations du signal [1, 8]. Cet article introduit ainsi une version à fenêtre glissante de l'algorithme API. La section 2 rappelle les principes de base de la méthode des puissances itérées. La section 3 introduit l'algorithme API à fenêtre glissante (SW-API), dont une implémentation rapide (SW-FAPI) est proposée dans la section 4. Enfin, les performances de cette méthode sont testées dans la section 5, et la section 6 résume les principaux résultats.

2 Méthode des puissances itérées

Soit $\{\boldsymbol{x}(t)\}_{t\geq 0}$ une séquence de vecteurs complexes de dimension n, dont la matrice de covariance $\boldsymbol{C}_{xx}(t)$ est estimée sur une fenêtre glissante de longueur l :

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

Cette matrice peut être mise à jour selon la récurrence

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

La méthode des puissances itérées [3] effectue la poursuite de l'espace dominant [1] de dimension $r \leq n$ engendré par la matrice $\boldsymbol{C}_{xx}(t)$. À chaque instant, elle détermine une base de cet espace, sous la forme d'une matrice $\boldsymbol{W}(t)$ orthonormée de dimension $n \times r$. Le calcul de $\boldsymbol{W}(t)$ comprend une étape de compression des données (3) et une étape d'orthonormalisation de la matrice compressée (4) :

1. L'espace dominant de dimension r engendré par la matrice hermitienne positive $\boldsymbol{C}_{xx}(t)$ est l'espace engendré par les r vecteurs propres de $\boldsymbol{C}_{xx}(t)$ associés aux r plus grandes valeurs propres.

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

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

L'étape d'orthonormalisation fait intervenir une matrice $\boldsymbol{R}(t)$, qui peut être vue comme une racine carrée de matrice[2]. En effet, l'orthonormalité de $\boldsymbol{W}(t)$ est équivalente à la relation $\boldsymbol{R}(t)^H\,\boldsymbol{R}(t) = \boldsymbol{\Phi}(t)$, où $\boldsymbol{\Phi}(t) \triangleq \boldsymbol{C}_{xy}(t)^H \boldsymbol{C}_{xy}(t)$. Il apparaît ainsi que $\boldsymbol{R}(t)^H$ est une racine carrée de $\boldsymbol{\Phi}(t)$, définie à une transformation unitaire près. On peut par exemple choisir $\boldsymbol{R}(t)$ triangulaire [5], ou hermitienne [3].

La multiplication dans l'étape (3) comporte $O(n^2 r)$ opérations, et l'orthonormalisation nécessite $O(nr^2)$ opérations. En raison de son coût élevé, cet algorithme n'est pas adapté pour un traitement temps réel, et nécessite d'être simplifié. Dans ce but, nous avons introduit dans [7] l'approximation (5), moins restrictive que celle connue sous le nom d'*approximation par projection*[3] [1] :

$$\boldsymbol{W}(t) \simeq \boldsymbol{W}(t-1)\,\boldsymbol{\Theta}(t) \tag{5}$$

où $\boldsymbol{\Theta}(t) \triangleq \boldsymbol{W}(t-1)^H \boldsymbol{W}(t)$. La méthode des puissances itérées peut être accélérée en introduisant l'approximation (5) à l'instant $t-1$ dans l'étape (3). On obtient ainsi une récurrence sur la matrice $\boldsymbol{C}_{xy}(t)$, qui permet une mise à jour rapide de la factorisation (4) entre les instants $t-1$ et t. Cette approche sera développée dans la section 3.

3 Algorithme SW-API

Commençons par établir la récurrence sur la matrice $\boldsymbol{C}_{xy}(t)$. Tout d'abord, l'équation (1) peut être réécrite sous la forme $\boldsymbol{C}_{xx}(t) = \boldsymbol{X}(t)\,\boldsymbol{X}(t)^H$, où $\boldsymbol{X}(t) \triangleq [\boldsymbol{x}(t-l+1), \ldots, \boldsymbol{x}(t)]$. Ainsi, l'équation (3) donne $\boldsymbol{C}_{xy}(t) = \boldsymbol{X}(t)\,\boldsymbol{Y}(t)^H$, où $\boldsymbol{Y}(t) \triangleq \boldsymbol{W}(t-1)^H \boldsymbol{X}(t)$. Il s'agit maintenant d'établir des récurrences sur $\boldsymbol{X}(t)$ et $\boldsymbol{Y}(t)$. La première est immédiate :

$$\left[\ \boldsymbol{x}(t-l)\ \middle|\ \boldsymbol{X}(t)\ \right] = \left[\ \boldsymbol{X}(t-1)\ \middle|\ \boldsymbol{x}(t)\ \right]. \tag{6}$$

En multipliant à gauche par $\boldsymbol{W}(t-1)^H$, on obtient

$$\left[\ \boldsymbol{v}(t-l)\ \middle|\ \boldsymbol{Y}(t)\ \right] = \left[\ \boldsymbol{W}(t-1)^H \boldsymbol{X}(t-1)\ \middle|\ \boldsymbol{y}(t)\ \right] \tag{7}$$

où $\boldsymbol{v}(t-l) \triangleq \boldsymbol{W}(t-1)^H \boldsymbol{x}(t-l)$ et $\boldsymbol{y}(t) \triangleq \boldsymbol{W}(t-1)^H \boldsymbol{x}(t)$. En appliquant l'approximation (5) à l'instant $t-1$ à l'équation (7), on obtient une récurrence sur $\boldsymbol{Y}(t)$:

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

où

$$\widehat{\boldsymbol{V}}(t-1) \triangleq \boldsymbol{\Theta}(t-1)^H \boldsymbol{Y}(t-1). \tag{8}$$

La définition exacte de $\boldsymbol{Y}(t)$ est alors remplacée par

$$\left[\ \widehat{\boldsymbol{v}}(t-l)\ \middle|\ \boldsymbol{Y}(t)\ \right] \triangleq \left[\ \widehat{\boldsymbol{V}}(t-1)\ \middle|\ \boldsymbol{y}(t)\ \right] \tag{9}$$

où le vecteur $\widehat{\boldsymbol{v}}(t-l)$, défini par la première colonne du membre de droite, est une approximation du vecteur $\boldsymbol{v}(t-l)$. Les équations (6), (8) et (9) impliquent finalement

$$\boldsymbol{C}_{xy}(t) = \boldsymbol{C}_{xy}(t-1)\boldsymbol{\Theta}(t-1) + \boldsymbol{x}(t)\boldsymbol{y}(t)^H - \boldsymbol{x}(t-l)\widehat{\boldsymbol{v}}(t-l)^H. \tag{10}$$

Nous souhaitons à présent utiliser l'équation (10) pour effectuer une mise à jour rapide de la factorisation (4). Ce calcul nécessite l'introduction d'une matrice auxiliaire, notée $\boldsymbol{Z}(t)$.

Soient $\boldsymbol{J} \triangleq \left[\begin{smallmatrix} 1 & 0 \\ 0 & -1 \end{smallmatrix}\right]$, $\underline{\boldsymbol{y}}(t) \triangleq \left[\ \boldsymbol{y}(t)\ \middle|\ \boldsymbol{v}(t-l)\ \right]$ et $\underline{\widehat{\boldsymbol{y}}}(t) \triangleq \left[\ \boldsymbol{y}(t)\ \middle|\ \widehat{\boldsymbol{v}}(t-l)\ \right]$. Par ailleurs, on suppose que la matrice $\boldsymbol{S}(t-1) \triangleq \left(\boldsymbol{R}(t-1)\boldsymbol{\Theta}(t-1)\right)^H$ est inversible. La proposition 1 introduit une récurrence sur $\boldsymbol{Z}(t-1) \triangleq \boldsymbol{S}(t-1)^{-1}$.

Proposition 1 *La matrice $\boldsymbol{S}(t) \triangleq \left(\boldsymbol{R}(t)\,\boldsymbol{\Theta}(t)\right)^H$ de dimension $r \times r$ est inversible si et seulement si la matrice $\boldsymbol{J} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)$ est inversible, où $\underline{\boldsymbol{h}}(t) \triangleq \boldsymbol{Z}(t-1)\,\underline{\widehat{\boldsymbol{y}}}(t)$. Dans ce cas, son inverse $\boldsymbol{Z}(t) \triangleq \boldsymbol{S}(t)^{-1}$ vérifie*

$$\boldsymbol{Z}(t) = \boldsymbol{\Theta}(t)^H \left(\boldsymbol{I}_r - \underline{\boldsymbol{g}}(t)\,\underline{\boldsymbol{y}}(t)^H\right) \boldsymbol{Z}(t-1)\,\boldsymbol{\Theta}(t)^{-H} \tag{11}$$

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

Preuve En substituant l'équation (4) dans l'équation (10) et en multipliant à gauche par $\boldsymbol{W}(t-1)^H$, on obtient

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

Considérons le lemme d'inversion matricielle [9, pp. 18-19]:

Lemme 1 *Soit $\boldsymbol{A}$ une matrice inversible de dimension $r \times r$. On définit la matrice $\boldsymbol{B} = \boldsymbol{A} + \boldsymbol{P}\,\boldsymbol{J}\,\boldsymbol{Q}$ de dimension $r \times r$, où les matrices $\boldsymbol{P}$, $\boldsymbol{Q}$ et $\boldsymbol{J}$ ont respectivement pour dimension $r \times m$, $m \times r$ et $m \times m$. On suppose de plus que $\boldsymbol{J}$ est inversible. Alors $\boldsymbol{B}$ est inversible si et seulement si $\boldsymbol{J}^{-1} + \boldsymbol{Q}\,\boldsymbol{A}^{-1}\,\boldsymbol{P}$ est inversible, et dans ce cas*

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

En appliquant le lemme 1 à l'équation (12) avec $m = 2$, on montre que la matrice $\boldsymbol{\Theta}(t)\,\boldsymbol{R}(t)$ est inversible si et seulement si $\boldsymbol{J} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)$ est inversible (ce qui fournit un moyen rapide de tester l'inversibilité commune de $\boldsymbol{\Theta}(t)$ et $\boldsymbol{R}(t)$). Dans ce cas, on obtient

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

Cette équation induit finalement la récurrence (11). □

Enfin, la proposition 2 introduit une récurrence sur $\boldsymbol{W}(t)$.

Proposition 2 *Dans le cas où la matrice $\boldsymbol{J} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)$ est inversible, $\boldsymbol{W}(t)$ satisfait la récurrence*

$$\boldsymbol{W}(t) = \left(\boldsymbol{W}(t-1) + \underline{\boldsymbol{e}}(t)\,\underline{\boldsymbol{g}}(t)^H\right) \boldsymbol{\Theta}(t) \tag{13}$$

où

$$\underline{\boldsymbol{e}}(t) \triangleq \underline{\boldsymbol{x}}(t) - \boldsymbol{W}(t-1)\,\underline{\boldsymbol{y}}(t). \tag{14}$$

Preuve En substituant l'équation (4) dans l'équation (10) et en multipliant à droite par $\boldsymbol{\Theta}(t)$, on montre que

$$\boldsymbol{W}(t)\boldsymbol{S}(t)^H = \left(\boldsymbol{W}(t-1)\boldsymbol{S}(t-1)^H + \underline{\boldsymbol{x}}(t)\boldsymbol{J}\underline{\widehat{\boldsymbol{y}}}(t)^H\right) \boldsymbol{\Theta}(t) \tag{15}$$

où $\underline{\boldsymbol{x}}(t) \triangleq \left[\ \boldsymbol{x}(t)\ \middle|\ \boldsymbol{x}(t-l)\ \right]$. En substituant les équations (12) et (14) dans l'équation (15), on obtient

$$\boldsymbol{W}(t)\boldsymbol{S}(t)^H = \boldsymbol{W}(t-1)\boldsymbol{\Theta}(t)\boldsymbol{S}(t)^H + \underline{\boldsymbol{e}}(t)\boldsymbol{J}\underline{\widehat{\boldsymbol{y}}}(t)^H\boldsymbol{\Theta}(t). \tag{16}$$

2. Si $\boldsymbol{T}$ est une matrice hermitienne positive, on appelle *racine carrée* de $\boldsymbol{T}$ toute matrice $\boldsymbol{S}$ de même dimension vérifiant l'égalité $\boldsymbol{S}\,\boldsymbol{S}^H = \boldsymbol{T}$, et on note $\boldsymbol{S} = \boldsymbol{T}^{\frac{1}{2}}$. Ainsi, les racines carrées sont définies à une transformation unitaire près, et la notation $\boldsymbol{S}^{\frac{1}{2}}$ peut désigner n'importe laquelle d'entre elles. Par contre, il n'existe qu'une seule racine carrée de $\boldsymbol{T}$ qui soit elle-même hermitienne positive.

3. Dans le cas de [1], $\boldsymbol{\Theta}(t)$ était contraint à être la matrice identité.

TAB. 1 – *Algorithme API à fenêtre glissante (SW-API)*

Initialisation :

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

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

À chaque instant faire

Section semblable à SW − PAST :

Vecteur d'entrée : $\boldsymbol{x}(t)$

$$\begin{bmatrix} \boldsymbol{x}(t-l) & \big| & \boldsymbol{X}(t) \end{bmatrix} = \begin{bmatrix} \boldsymbol{X}(t-1) & \big| & \boldsymbol{x}(t) \end{bmatrix}$$

$\boldsymbol{y}(t) = \boldsymbol{W}(t-1)^H \boldsymbol{x}(t)$ — $O(nr)$

$$\begin{bmatrix} \widehat{\boldsymbol{v}}(t-l) & \big| & \boldsymbol{Y}(t) \end{bmatrix} = \begin{bmatrix} \widehat{\boldsymbol{V}}(t-1) & \big| & \boldsymbol{y}(t) \end{bmatrix}$$

$\boldsymbol{v}(t-l) = \boldsymbol{W}(t-1)^H \boldsymbol{x}(t-l)$ — $O(nr)$

$$\underline{\boldsymbol{x}}(t) = \begin{bmatrix} \boldsymbol{x}(t) & \big| & \boldsymbol{x}(t-l) \end{bmatrix}$$

$$\underline{\widehat{\boldsymbol{y}}}(t) = \begin{bmatrix} \boldsymbol{y}(t) & \big| & \widehat{\boldsymbol{v}}(t-l) \end{bmatrix}$$

$$\underline{\boldsymbol{y}}(t) = \begin{bmatrix} \boldsymbol{y}(t) & \big| & \boldsymbol{v}(t-l) \end{bmatrix}$$

$\underline{\boldsymbol{h}}(t) = \boldsymbol{Z}(t-1)\,\underline{\widehat{\boldsymbol{y}}}(t)$ — $O(r^2)$

$\underline{\boldsymbol{g}}(t) = \underline{\boldsymbol{h}}(t)\left(\boldsymbol{J} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)\right)^{-1}$ — $O(r)$

$\underline{\boldsymbol{e}}(t) = \underline{\boldsymbol{x}}(t) - \boldsymbol{W}(t-1)\,\underline{\boldsymbol{y}}(t)$ — $O(nr)$

Section principale de SW − API :

$\boldsymbol{\Theta}(t) = \left(\boldsymbol{I}_r + \underline{\boldsymbol{g}}(t)\left(\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)\right)\underline{\boldsymbol{g}}(t)^H\right)^{-\frac{1}{2}}$ — $O(n+r^3)$

$\boldsymbol{Z}(t) = \boldsymbol{\Theta}(t)^H\left(\boldsymbol{I}_r - \underline{\boldsymbol{g}}(t)\,\underline{\boldsymbol{y}}(t)^H\right)\boldsymbol{Z}(t-1)\,\boldsymbol{\Theta}(t)^{-H}$ — $O(r^3)$

$\boldsymbol{W}(t) = \left(\boldsymbol{W}(t-1) + \underline{\boldsymbol{e}}(t)\,\underline{\boldsymbol{g}}(t)^H\right)\boldsymbol{\Theta}(t)$ — $O(nr^2)$

$\widehat{\boldsymbol{V}}(t) = \boldsymbol{\Theta}(t)^H \boldsymbol{Y}(t)$ — $O(lr^2)$

Or en multipliant l'équation (12) à gauche par $\underline{\boldsymbol{g}}(t)^H$, on a

$$\boldsymbol{J}\,\underline{\widehat{\boldsymbol{y}}}(t)^H = \underline{\boldsymbol{g}}(t)^H\,\boldsymbol{\Theta}(t)\,\boldsymbol{R}(t). \tag{17}$$

En substituant l'équation (17) dans (16) et en multipliant à droite par $\boldsymbol{Z}(t)^H$, on obtient finalement (13). □

Puisque la matrice $\boldsymbol{W}(t-1)$ est orthonormale, le vecteur $\underline{\boldsymbol{e}}(t)$ est orthogonal à $\boldsymbol{W}(t-1)$. Par conséquent, l'orthonormalité de $\boldsymbol{W}(t)$, conjuguée à l'équation (13), se traduit par

$$\boldsymbol{\Theta}(t)\,\boldsymbol{\Theta}(t)^H = \left(\boldsymbol{I}_r + \underline{\boldsymbol{g}}(t)\,\left(\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)\right)\,\underline{\boldsymbol{g}}(t)^H\right)^{-1}. \tag{18}$$

Ainsi, $\boldsymbol{\Theta}(t)$ est une racine carrée inverse de la matrice hermitienne $\boldsymbol{I}_r + \underline{\boldsymbol{g}}(t)\,\left(\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)\right)\,\underline{\boldsymbol{g}}(t)^H$. Le choix d'une racine particulière n'affectera pas la performance du suivi. Le pseudo-code complet de l'algorithme SW-API est donné dans la table 1. La première section ressemble à l'algorithme SW-PAST, tel que présenté dans l'article [8]. Sa complexité est $O(nr)$, alors que celle de la seconde section est $O((n+l)r^2)$.

Dans le cas particulier où la matrice $\boldsymbol{J} + \underline{\boldsymbol{y}}(t)^H \underline{\boldsymbol{h}}(t)$ n'est pas inversible, $\boldsymbol{Z}(t)$ et $\boldsymbol{W}(t)$ ne peuvent plus être mis à jour avec les équations (11) et (13), mais une solution a été proposée dans [7]. En pratique, nous n'avons jamais rencontré un tel cas dans nos simulations numériques.

4 Algorithme SW-API rapide

Dans cette section, nous proposons une implémentation rapide de l'algorithme SW-API, baptisée SW-FAPI, reposant sur un choix particulier de la matrice $\boldsymbol{\Theta}(t)$, qui réduit la complexité globale à $O((n+l)r)$. Soit $\underline{e}(t)$ une racine carrée de la matrice

TAB. 2 – *Algorithme SW-API rapide (SW-FAPI)*

Initialisation (*cf.* table 1)

À chaque instant faire

Section semblable à SW − PAST (*cf.* table 1)

Section principale de SW − FAPI :

$\underline{e}(t) = \left(\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)\right)^{\frac{1}{2}}$ — $O(r)$

$\underline{\rho}(t) = I_2 + \underline{e}(t)^H\left(\underline{\boldsymbol{g}}(t)^H \underline{\boldsymbol{g}}(t)\right)\underline{e}(t)$ — $O(r)$

$\underline{\tau}(t) = \underline{e}(t)\left(\rho(t) + \rho(t)^{\frac{H}{2}}\right)^{-1}\underline{e}(t)^H$ — $O(1)$

$\underline{\eta}(t) = I_2 - \left(\underline{\boldsymbol{g}}(t)^H \underline{\boldsymbol{g}}(t)\right)\underline{\tau}(t)$ — $O(1)$

$\underline{\boldsymbol{h}}'(t) = \boldsymbol{Z}(t-1)^H\left(\underline{\boldsymbol{y}}(t)\,\underline{\eta}(t) + \underline{\boldsymbol{g}}(t)\,\underline{\tau}(t)\right)$ — $O(r^2)$

$\underline{\boldsymbol{\epsilon}}(t) = \Big(\boldsymbol{Z}(t-1)\underline{\boldsymbol{g}}(t) - \underline{\boldsymbol{g}}(t)\left(\underline{\boldsymbol{h}}'(t)^H \underline{\boldsymbol{g}}(t)\right)\Big) \times \left(\underline{\tau}(t)\underline{\eta}(t)^{-1}\right)^H$ — $O(r^2)$

$\boldsymbol{Z}(t) = \boldsymbol{Z}(t-1) - \underline{\boldsymbol{g}}(t)\,\underline{\boldsymbol{h}}'(t)^H + \underline{\boldsymbol{\epsilon}}(t)\,\underline{\boldsymbol{g}}(t)^H$ — $O(r^2)$

$\underline{\boldsymbol{e}}'(t) = \underline{\boldsymbol{e}}(t)\,\underline{\eta}(t) - \boldsymbol{W}(t-1)\,\underline{\boldsymbol{g}}(t)\,\underline{\tau}(t)$ — $O(nr)$

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

$\widehat{\boldsymbol{V}}(t) = \boldsymbol{Y}(t) - \underline{\boldsymbol{g}}(t)\,\underline{\tau}(t)^H\left(\underline{\boldsymbol{g}}(t)^H \boldsymbol{Y}(t)\right)$ — $O(lr)$

$\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)$, et $\underline{\rho}(t) = I_2 + \underline{e}(t)^H\left(\underline{\boldsymbol{g}}(t)^H \underline{\boldsymbol{g}}(t)\right)\underline{e}(t)$. On définit alors la matrice 2×2

$$\underline{\tau}(t) = \underline{e}(t)\left(\underline{\rho}(t) + \underline{\rho}(t)^{\frac{H}{2}}\right)^{-1}\underline{e}(t)^H. \tag{19}$$

Même si d'autres choix seraient possibles, nous supposerons dans la suite que la racine carrée de $\underline{\rho}(t)$ qui intervient dans l'équation (19) est l'unique racine hermitienne définie positive. Cette condition garantit que la matrice $\underline{\rho}(t) + \underline{\rho}(t)^{\frac{H}{2}}$ est bien inversible, et que la matrice $\underline{\tau}(t)$ ainsi définie est elle-même hermitienne positive. On peut alors vérifier que

$$\boldsymbol{\Theta}(t) \triangleq \boldsymbol{I}_r - \underline{\boldsymbol{g}}(t)\,\underline{\tau}(t)\,\underline{\boldsymbol{g}}(t)^H \tag{20}$$

est une racine carrée inverse de $\boldsymbol{I}_r + \underline{\boldsymbol{g}}(t)\,\left(\underline{\boldsymbol{e}}(t)^H \underline{\boldsymbol{e}}(t)\right)\,\underline{\boldsymbol{g}}(t)^H$. En particulier, si $\tau(t)$ est à symétrie hermitienne, $\boldsymbol{\Theta}(t)$ présente la même symétrie. Posons ensuite

$$\underline{\eta}(t) = I_2 - \left(\underline{\boldsymbol{g}}(t)^H \underline{\boldsymbol{g}}(t)\right)\underline{\tau}(t). \tag{21}$$

Puisque $\boldsymbol{\Theta}(t)$ est inversible, le lemme d'inversion matricielle montre que $\underline{\eta}(t)$ l'est aussi[4]. En substituant l'équation (20) dans l'équation (11), on obtient

$$\boldsymbol{Z}(t) = \boldsymbol{Z}(t-1) - \underline{\boldsymbol{g}}(t)\,\underline{\boldsymbol{h}}'(t)^H + \underline{\boldsymbol{\epsilon}}(t)\,\underline{\boldsymbol{g}}(t)^H \tag{22}$$

où

$$\underline{\boldsymbol{h}}'(t) = \boldsymbol{Z}(t-1)^H\left(\underline{\boldsymbol{y}}(t)\,\underline{\eta}(t) + \underline{\boldsymbol{g}}(t)\,\underline{\tau}(t)\right) \tag{23}$$

et

$$\underline{\boldsymbol{\epsilon}}(t) = \left(\boldsymbol{Z}(t-1)\,\underline{\boldsymbol{g}}(t) - \underline{\boldsymbol{g}}(t)\,\left(\underline{\boldsymbol{h}}'(t)^H \underline{\boldsymbol{g}}(t)\right)\right)\left(\underline{\tau}(t)\,\underline{\eta}(t)^{-1}\right)^H. \tag{24}$$

Par ailleurs, en substituant l'équation (20) dans l'équation (13), on obtient

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

où

$$\underline{\boldsymbol{e}}'(t) = \underline{\boldsymbol{e}}(t)\,\underline{\eta}(t) - \boldsymbol{W}(t-1)\,\underline{\boldsymbol{g}}(t)\,\underline{\tau}(t). \tag{26}$$

Enfin, en substituant l'équation (20) dans (8), on a

$$\widehat{\boldsymbol{V}}(t) = \boldsymbol{Y}(t) - \underline{\boldsymbol{g}}(t)\,\underline{\tau}(t)^H\left(\underline{\boldsymbol{g}}(t)^H \boldsymbol{Y}(t)\right). \tag{27}$$

L'algorithme SW-FAPI est résumé dans la table 2.

4. On applique le lemme 1 à l'équation (20), en posant $\boldsymbol{A} = \boldsymbol{I}_r$, $\boldsymbol{P} = \underline{\boldsymbol{g}}(t)\,\underline{\tau}(t)$, $\boldsymbol{J} = I_2$ et $\boldsymbol{Q} = \underline{\boldsymbol{g}}(t)^H$.

5 Simulations numériques

Dans cette section, les algorithmes de suivi d'espace dominant sont appliqués dans le contexte de l'estimation de fréquences. Les performances du suivi sont mesurées en terme d'angle maximal entre sous-espaces [4, pp. 603-604]). À chaque instant t, on compare l'espace dominant exact de la matrice de covariance $\boldsymbol{C}_{xx}(t)$ à l'espace estimé par l'algorithme de suivi. Le signal de test est une somme de $r = 4$ sources sinusoïdales complexes et d'un bruit blanc gaussien complexe (le rapport signal à bruit est de 5.7 dB). Les fréquences des sinusoïdes varient par paliers. Leurs variations sont représentées sur la figure 1-a. La figure 1-b représente les variations temporelles de l'angle maximal d'erreur $\theta_{\mathrm{SW-FAPI}}(t)$ obtenu avec la méthode SW-FAPI[5]. On peut voir que cet algorithme converge rapidement après chaque saut de fréquence. Ce résultat peut être comparé à celui de la figure 1-c, obtenu avec la méthode FAPI à oubli exponentiel[6] [7]. On observe que la réponse aux sauts de fréquence est plus lente, en raison de la nature de la fenêtre d'analyse, qui tend à lisser les variations du signal. Nous avons également pu constater que l'algorithme SW-FAPI donne des résultats très proches de ceux de l'algorithme SW-OPAST [8] et de l'algorithme NIC [2], qui constitue une généralisation robuste de PAST, et dont nous avons implémenté une version à fenêtre glissante SW-NIC[7]. Les performances de ces divers algorithmes sont comparées dans les figures 1-d et 1-e. La figure 1-d représente le rapport en décibels des angles maximaux d'erreur obtenus avec les algorithmes SW-FAPI et SW-NIC, c'est-à-dire $20\log_{10}\left(\frac{\theta_{\mathrm{SW-FAPI}}(t)}{\theta_{\mathrm{SW-NIC}}(t)}\right)$, et la figure 1-e représente le rapport des angles d'erreur obtenus avec SW-FAPI et SW-OPAST. On peut ainsi observer que SW-FAPI s'avère globalement plus précis que les algorithmes SW-OPAST et SW-NIC. Finalement, l'orthonormalité de la matrice $\boldsymbol{W}(t)$ peut être mesurée en décibels à l'aide du critère d'erreur $20\log_{10}(\|\boldsymbol{W}(t)^H\boldsymbol{W}(t) - \boldsymbol{I}_r\|_F)$. Sur le signal de test, SW-NIC atteint une erreur de -10 dB, alors que les algorithmes SW-OPAST, SW-FAPI et FAPI ne dépassent jamais -240, -285 et -305 dB respectivement.

6 Conclusion

Dans cet article, nous avons présenté une version à fenêtre glissante de l'algorithme API. Cet algorithme garantit l'orthonormalité de la matrice générée à chaque instant. De plus, sa complexité est seulement $O((n+l)r)$, et il hérite de la propriété de convergence globale et exponentielle vérifiée par la méthode des puissances itérées. Dans le contexte de l'estimation de fréquences, cette technique s'est avérée robuste à de brusques variations du signal, et permet d'atteindre des performances supérieures à celles de l'algorithme OPAST à fenêtre glissante.

5. Les dimensions de la matrice de données ont été fixées à $n = 80$ et $l = 120$.

6. Le facteur d'oubli $\alpha \simeq 0.99$ a été choisi pour obtenir une longueur effective de fenêtre égale à l.

7. Le pas d'apprentissage η a été fixé à 0.5.

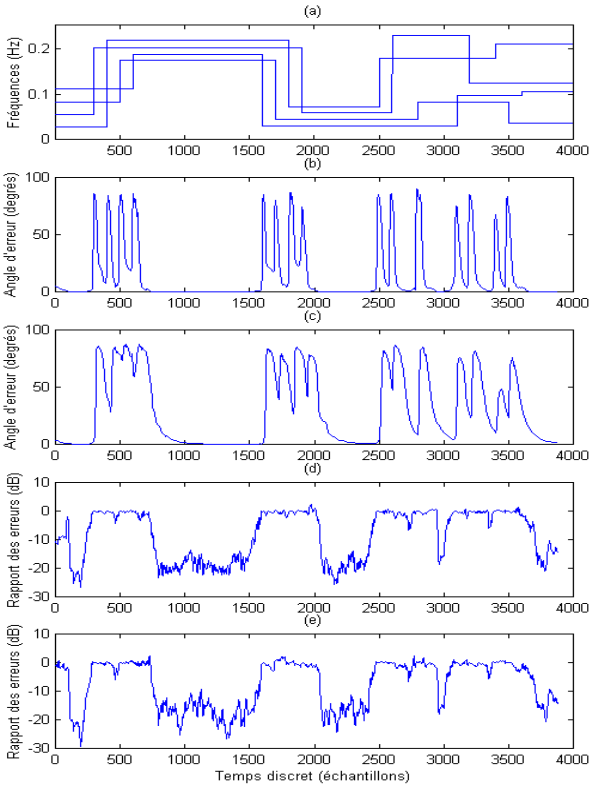

FIG. 1 – *Résultat des simulations numériques*

(a) Fréquences réduites des sinusoïdes
(b) Trajectoire d'erreur avec SW-FAPI
(c) Trajectoire d'erreur avec FAPI
(d) Rapport des erreurs obtenues avec SW-FAPI et SW-NIC
(e) Rapport des erreurs obtenues avec SW-FAPI et SW-OPAST

Références

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

[2] Y. Miao et Y. Hua, "Fast subspace tracking and neural network learning by a novel information criterion," *IEEE Trans. on Signal Proc.*, vol. 46, no. 7, jui. 1998.

[3] Y. Hua, Y. Xiang, T. Chen, K. Abed-Meraim, et Y. Miao, "A new look at the power method for fast subspace tracking," *Digital Signal Proc.*, oct. 1999.

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

[5] P. Strobach, "Low-rank adaptive filters," *IEEE Trans. on Signal Proc.*, vol. 44, no. 12, déc. 1996.

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

[7] R. Badeau, G. Richard, B. David, et K. Abed-Meraim, "Approximated power iterations for fast subspace tracking," dans *Proc. ISSPA 2003*, jui. 2003.

[8] R. Badeau, K. Abed-Meraim, G. Richard, et B. David, "Sliding window orthonormal PAST algorithm," dans *Proc. IEEE ICASSP 2003*, avr. 2003.

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