Algorithmes de poursuite de sous-espace avec contrainte de parcimonie

Nacerredine Lassami, Karim Abed-Meraim, Abdeldjalil Aissa El Bey

Algorithmes de poursuite de sous-espace avec contrainte de parcimonie

Nacerredine LASSAMI[1], Karim ABED-MERAIM[2], Abdeldjalil AÏSSA-EL-BEY[1]

[1]IMT Atlantique, UMR CNRS 6285 Lab-STICC, Université Bretagne Loire, 29238 Brest, France

[2]Laboratoire PRISME, Université d'Orléans, 12 Rue de Blois, 45067 Orléans, France

nacerredine.lassami@imt-atlantique.fr, karim.abed-meraim@univ-orleans.fr
abdeldjalil.aissaelbey@imt-atlantique.fr

Résumé – Dans cet article, nous considérons la poursuite du sous-espace principal (signal) sous la contrainte de parcimonie de la matrice de poids. Plus précisément, nous proposons une approche en deux étapes où la première utilise l'algorithme FAPI pour une extraction adaptative d'une base orthonormée du sous-espace principal. Une estimation de la matrice de poids souhaitée est effectuée dans la deuxième étape, en tenant compte de la contrainte de parcimonie. Les algorithmes résultants: SS-FAPI et SS-FAPI2 ont une faible complexité de calcul (adaptée au contexte adaptatif) et ils réalisent à la fois de bonnes performances en convergence et en estimation, comme le montrent nos résultats de simulation.

Abstract – In this article, we consider the principal (signal) subspace tracking under a sparsity constraint on the weight matrix. More precisely, we propose a two-step approach to solve the problem. The first step uses the FAPI algorithm for adaptive extraction of an orthonormal basis of the principal subspace. An estimation of the desired weight matrix is carried out in the second step, taking into account the sparsity constraint. The resulting algorithms: SS-FAPI and SS-FAPI2 have a low computational complexity (suitable for the adaptive context) and they achieve both good convergence and estimation performance as illustrated by our simulation results.

1 Introduction

Les techniques sous-espaces jouent un rôle fondamental en estimation statistique et en traitement d'antennes. En effet, elles sont largement utilisées dans des applications telle que la compression de données, l'identification de système, le filtrage et l'estimation de paramètres. L'estimation du sous-espace signal a pour but d'estimer une matrice qu'on appelle la matrice de poids et qui engendre le même sous-espace que les vecteurs propres dominants de la matrice de covariance des données. Généralement, l'estimation est basée sur la décomposition en valeurs propres (EVD) ou la décomposition en valeurs singulières (SVD) d'une statistique des données. Toutefois, le principal inconvénient de ces décompositions est leur forte complexité numérique qui les rends inappropriées dans le cas où la non stationnarité du système nécessite des mises à jour régulières de ces grandeurs. Il existe donc un réel besoin de suivi rapide des sous-espaces dans le contexte du traitement adaptatif des signaux. De nombreux algorithmes ont été proposés pour résoudre ce problème, notamment la méthode Oja [1] dont il a été établi dans [2] qu'elle peut être vue comme une technique de gradient approximée pour la minimisation d'une certaine erreur quadratique moyenne. Par la suite, d'autres algorithmes plus rapides ont été proposés tels que : PAST [2] et sa variante orthogonale OPAST [3]. La poursuite du sous-espace peut être aussi basée sur l'algorithme de puissance itéré, c'est le cas de LORAF2 [4] et FAPI [5][6]. Plus récemment, certaines applications nécessitent de résoudre le problème de sous-espace principal mais sous une contrainte de parcimonie sur la matrice de poids, comme dans le cas de ACP (l'analyse en composantes principales) parcimonieuses [7] où la parcimonie facilite l'interprétation et améliore le comportement en grande dimension de la ACP classique. Étonnamment, bien que plusieurs solutions aient été mises en place pour l'estimation des sous-espaces en "batch" (toutes les données sont disponibles à la fois), il n'existe que peu de solutions jusqu'à présent pour le cas des systèmes adaptatifs. Parmi les solutions adaptatives existantes, on trouve l'application [8] proposée dans le cadre du STAP (space-time adaptive processing) pour le radar aéroporté à phase progressive, où l'algorithme noté ℓ_1-PAST résout le problème de poursuite du sous-espace principal sous la contrainte de parcimonie sur la matrice de poids.

Dans cet article, on s'intéresse à résoudre le même problème, mais sans prendre en compte la contrainte d'orthogonalité. En effet, on a constaté que la recherche du meilleur sous-espace orthogonal et le plus parcimonieux en même temps n'est pas toujours judicieuse et peut conduire à des solutions sous-optimales. Nous proposons un algorithme en deux étapes qui a la même performance sous-espace que FAPI tout en fournissant la matrice de poids la plus parcimonieuse qui engendre ce sous-espace principal. D'autres versions algorithmiques sont aussi étudiées et ce afin d'améliorer les performances du point de vue de la qualité d'estimation du sous-espace ou du point de vue de la réduction de la complexité numérique.

2 Formulation du problème

Soit $\mathbf{x}(t) \in \mathbb{R}^n$ un vecteur de données aléatoires observé sur n capteurs à l'instant t, et soit $\mathbf{C}_{xx} = E[\mathbf{x}(t)\mathbf{x}(t)^T]$ sa matrice de covariance. Dans un contexte adaptatif, on utilise une fenêtre exponentielle pour la mise à jour de la matrice de covariance :

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

où $0 < \beta < 1$ est un facteur d'oubli utilisé pour permettre la poursuite dans le cas non-stationnaire. Considérons le problème d'estimation du sous-espace principal engendré par la séquence $\{\mathbf{x}(t)\}$, de dimension $p < n$, qui coïncide théoriquement avec l'espace des p vecteurs propres dominants de la matrice de covariance $\mathbf{C}_{xx}$ avec une contrainte de parcimonie sur la matrice de poids $\mathbf{W}(t) \in \mathbb{R}^{n\times p}$ ($\mathbf{W}(t)$ est la matrice qui génère le sous espace désiré). Pour résoudre ce problème, on propose une approche à deux étapes où la première nous permet de calculer une base orthonormale du sous-espace principal de $\mathbf{C}_{xx}$ et la deuxième est dédiée à l'estimation de la matrice de poids parcimonieuse. Dans la première étape, on optimise :

$$J_{PS}(\mathbf{W}(t)) = \sum_{i=1}^{t} \beta^{t-i}\|\mathbf{x}(i) - \mathbf{W}(t)\mathbf{W}^T(t)\mathbf{x}(i)\|_2^2 \quad (1)$$

sous la contrainte d'orthonormalité $\mathbf{W}^T(t)\mathbf{W}(t) = \mathbf{I}_p$. Cette fonction [1] représente l'erreur quadratique moyenne du vecteur de données projetées $(\mathbf{I}_n - \mathbf{W}(t)\mathbf{W}^T(t))\mathbf{x}(t)$. L'optimisation est réalisée avec l'algorithme FAPI [5][6] (décrit en section 3.1) et le résultat est noté par $\mathbf{W}_{PS}(t)$. Dans la deuxième étape, on cherche la matrice de poids parcimonieuse $\mathbf{W}(t)$ sous la forme $\mathbf{W}(t) = \mathbf{W}_{PS}(t)\mathbf{Q}(t)$ où la matrice inversible $\mathbf{Q}(t) \in \mathbb{R}^{p\times p}$ est calculée en optimisant la fonction [2] objectif :

$$J_{SS}(\mathbf{Q}(t)) = \|\mathbf{W}(t)\|_1 = \|\mathbf{W}_{PS}(t)\mathbf{Q}(t)\|_1 \quad (2)$$

sous la contrainte $\sum_{i=1}^{n} \mathbf{W}_{ij}(t)^2 = 1$ pour $j = 1, ..., p$ avec $\|\mathbf{W}(t)\|_1 \triangleq \sum_{i=1}^{n}\sum_{j=1}^{p}|\mathbf{W}_{ij}(t)|$. La pseudo norme ℓ_0 est plus adéquate pour représenter la parcimonie de $\mathbf{W}(t)$ mais ceci rend la fonction objectif non convexe et difficile à optimiser. Par conséquence,on utilise la relaxation en norme ℓ_1 qui est l'une des meilleures approximations convexes de ℓ_0.

• **Remarque :** Soulignons que l'objectif d'orthogonalité de la matrice de poids $\mathbf{W}$ n'est pas forcément aligné à celui de sa parcimonie, ce qui nous a conduit à considérer une approche à deux étapes avec deux matrices $\mathbf{W}_{PS}(t)$ et $\mathbf{W}(t)$. Aussi, la forme choisie pour $\mathbf{W}(t)$ fait que l'on préserve le sous-espace estimé et ce afin de maintenir la qualité de poursuite de FAPI.

3 La poursuite du sous-espace

Dans cette section, on présente les différents algorithmes utilisés dans la poursuite du sous-espace principal parcimonieux. Nous commençons par rappeler l'algorithme FAPI [5][6].

1. L'indice "PS"correspond à :"Principal Subspace"
2. L'indice "SS"correspond à :"Sparse System matrix "

3.1 Algorithme FAPI

La méthode de puissance itérée classique est basée sur le calcul de $\mathbf{C}_{xy}(t)$ et sa décomposition QR comme suit :

$$\mathbf{C}_{xy}(t) = \mathbf{C}_{xx}(t)\mathbf{W}(t-1) \quad \text{et} \quad \mathbf{W}(t)\mathbf{R}(t) = \mathbf{C}_{xy}(t) \quad (3)$$

où $\mathbf{C}_{xy}(t) \in \mathbb{R}^{n\times p}$ est la matrice de corrélation entre le vecteur des données $\mathbf{x}(t)$ et le vecteur des données projetés $\mathbf{y}(t) = \mathbf{W}(t-1)^T\mathbf{x}(t)$. On suppose que les matrices orthonormales $\mathbf{W}(t)$ et $\mathbf{W}(t-1)$ engendrent approximativement le même sous-espace que les p vecteurs propres dominants de $\mathbf{C}_{xy}(t)$, ce qui nous permet d'écrire :

$$\mathbf{W}(t) \approx \mathbf{W}(t-1)\mathbf{\Theta}(t) \quad \text{et} \quad \mathbf{R}^T(t) \approx \mathbf{C}_{yy}(t)\mathbf{\Theta}(t) \quad (4)$$

avec $\mathbf{\Theta}(t) \triangleq \mathbf{W}(t-1)^T\mathbf{W}(t)$ une matrice orthonormale et $\mathbf{C}_{yy}(t)$ est la matrice d'auto-corrélation du vecteur $\mathbf{y}(t)$. Utilisant (4) ainsi que l'approximation de projection proposée dans [2], conduit à l'algorithme suivant (voir détails dans [5][6]) :

Algorithme 1 FAPI

$\mathbf{y}(t) = \mathbf{W}(t-1)^T\mathbf{x}(t)$
$\mathbf{h}(t) = \mathbf{Z}(t-1)\mathbf{y}(t)$
$\mathbf{g}(t) = \frac{\mathbf{h}(t)}{\beta+\mathbf{y}(t)^T\mathbf{h}(t)}$
$\varepsilon^2(t) = \|\mathbf{x}(t)\|^2 - \|\mathbf{y}(t)\|^2$
$\tau(t) = \frac{\varepsilon^2(t)}{1+\varepsilon^2(t)\|\mathbf{g}(t)\|^2+\sqrt{1+\varepsilon^2(t)\|\mathbf{g}(t)\|^2}}$
$\eta(t) = 1 - \tau(t)\|\mathbf{g}(t)\|^2$
$\mathbf{y}'(t) = \eta(t)\mathbf{y}(t) + \tau(t)\mathbf{g}(t)$
$\mathbf{h}'(t) = \mathbf{Z}(t-1)^T\mathbf{y}'(t)$
$\boldsymbol{\epsilon}(t) = \frac{\tau(t)}{\eta(t)}(\mathbf{Z}(t-1)\mathbf{g}(t) - (\mathbf{h}'(t)^T\mathbf{g}(t))\mathbf{g}(t))$
$\mathbf{Z}(t) = \frac{1}{\beta}(\mathbf{Z}(t-1) - \mathbf{g}(t)\mathbf{h}'(t)^T + \boldsymbol{\epsilon}(t)\mathbf{g}(t)^T)$
$\mathbf{e}'(t) = \eta(t)\mathbf{x}(t) - \mathbf{W}(t-1)\mathbf{y}'(t)$
$\mathbf{W}(t) = \mathbf{W}(t-1) + \mathbf{e}'(t)\mathbf{g}(t)^T$

3.2 Algorithmes de complexité $O(np^2)$

En se basant sur la base orthonormale du sous-espace estimée par FAPI [5][6] de complexité $O(np)$ (ou par LORAF2 [4] de complexité $O(np^2)$), nous allons optimiser la fonction J_{SS} en utilisant une méthode de gradient naturel. On cherche maintenant la matrice $\mathbf{Q}(t) \in \mathbb{R}^{p\times p}$ inversible qui minimise $\|\mathbf{W}_{PS}(t)\mathbf{Q}(t)\|_1$ i.e. elle transforme la base orthonormale $\mathbf{W}_{PS}(t)$ en une base parcimonieuse. Ainsi, nous allons mettre à jour $\mathbf{Q}(t)$ sous la forme $\mathbf{Q}(t) = \mathbf{Q}(t-1)(\mathbf{I}_p + \boldsymbol{\epsilon})$ où la matrice $\boldsymbol{\epsilon} \in \mathbb{R}^{p\times p}$ a des valeurs faibles qui peuvent être calculées en utilisant une approximation du premier ordre :

$$\hat{\boldsymbol{\epsilon}} = \arg\min_{\boldsymbol{\epsilon}} \|\mathbf{W}_{PS}(t)\mathbf{Q}(t-1) + \mathbf{W}_{PS}(t)\mathbf{Q}(t-1)\boldsymbol{\epsilon}\|_1 \quad (5)$$

$$\hat{\boldsymbol{\epsilon}} = \arg\min_{\boldsymbol{\epsilon}} \sum_{i=1}^{n}\sum_{j=1}^{p}|\mathbf{M}_{ij} + \sum_{k=1}^{p}\mathbf{M}_{ik}\boldsymbol{\epsilon}_{kj}| \quad (6)$$

avec $\mathbf{M} = \mathbf{W}_{PS}(t)\mathbf{Q}(t-1)$. Sous l'hypothèse $|z| << |x|$, on a $|x+z| = |x| + \text{sign}(x)z$. On utilise ce résultat dans (6), on

trouve :

$$\hat{\boldsymbol{\epsilon}} \approx \arg\min_{\boldsymbol{\epsilon}} \sum_{i=1}^{n}\sum_{j=1}^{p} |\mathbf{M}_{ij}| + \text{sign}(\mathbf{M}_{ij}) \sum_{k=1}^{p} \mathbf{M}_{ik}\boldsymbol{\epsilon}_{kj} \quad (7)$$

$$\hat{\boldsymbol{\epsilon}} \approx \arg\min_{\boldsymbol{\epsilon}} \|\mathbf{M}\|_1 + Tr(\boldsymbol{\epsilon}\mathbf{D}^T) \quad (8)$$

avec $\mathbf{D} = \mathbf{M}^T\text{sign}(\mathbf{M})$ où $\text{sign}(\mathbf{M})$ est la matrice qui a comme élément le signe de $\mathbf{M}_{ij}$. Alors en choisissant $\hat{\boldsymbol{\epsilon}} = -\mu\frac{\mathbf{D}}{\|\mathbf{D}\|^2}$ avec $\mu > 0$, nous assurons la décroissance local de la fonction objectif selon : $\|\mathbf{M}\|_1 - \mu < \|\mathbf{M}\|_1$ (Le choix de μ est discuté dans la section 3.4). Après chaque itération et pour bien contrôler le conditionnement de $\mathbf{W}(t)$, on normalise les colonnes de $\mathbf{Q}(t)$ comme résumé dans le tableau ci-dessous.

Algorithme 2 SS-FAPI

1: Calculer $\mathbf{W}_{PS}(t)$ en utilisant FAPI
2: $\mathbf{M} = \mathbf{W}_{PS}(t)\mathbf{Q}(t-1)$
3: $\mathbf{D} = \mathbf{M}^T\text{sign}(\mathbf{M})$ ensuite $\mathbf{D} = \frac{\mathbf{D}}{\|\mathbf{D}\|^2}$
4: $\mathbf{Q}(t) = \mathbf{Q}(t-1)(\mathbf{I}_p - \mu\mathbf{D})$
5: Normaliser les colonnes de $\mathbf{Q}(t)$
6: $\mathbf{W}(t) = \mathbf{W}_{PS}(t)\mathbf{Q}(t)$.

La complexité de notre algorithme est de l'ordre $O(np^2)$ à cause de l'optimisation ℓ_1. Nous avons donc deux directions d'améliorations possibles : soit en réduisant le coût de l'optimisation ℓ_1 et réduire la complexité globale à $O(np)$, soit utiliser un algorithme de poursuite du sous-espace plus performant que FAPI mais de complexité $O(np^2)$ tout en gardant la complexité globale $O(np^2)$. Ainsi, en suivant cette dernière direction, nous avons créé une variante de SS-FAPI en utilisant l'algorithme LORAF2 [4] et on la note SS-LORAF2.

3.3 Algorithmes de complexité $O(np)$

Sachant que c'est la minimisation ℓ_1 qui rend la complexité de SS-FAPI d'ordre $O(np^2)$, on a cherché à réduire le coût de calcul de celle-ci. L'idée est de ne pas utiliser toute la matrice $\mathbf{D} = \mathbf{M}^T\text{sign}(\mathbf{M})$ pour mettre à jour $\mathbf{Q}(t)$, mais de ne garder que deux éléments de $\mathbf{D}$ ayant (i,j) et (j,i) comme cordonnées. Le choix de i et j se fait automatiquement en balayant toutes les paires possibles au cours des itérations avec $1 \leq i < j \leq p$. Pour le calcul de d_{ij} et d_{ji}, on n'aura besoin que de calculer les deux colonnes $\mathbf{M}_i$ et $\mathbf{M}_j$ de la matrice $\mathbf{M}$:

$$\mathbf{M}_i = \mathbf{W}_i(t-1) + \tilde{g}_i\mathbf{e}' \quad \text{et} \quad \mathbf{M}_j = \mathbf{W}_j(t-1) + \tilde{g}_j\mathbf{e}' \quad (9)$$

où $\mathbf{W}_j(t-1)$ est la $j-eme$ colonne de la base parcimonieuse à l'itération précédente et $\tilde{g}_j$ est le $j-eme$ élément du vecteur $\tilde{\mathbf{g}} = \mathbf{Q}(t-1)^T\mathbf{g}$ ($\mathbf{g}$ et $\mathbf{e}'$ sont issus de la dernière itération de FAPI). Ensuite, on calcule les deux éléments d_{ij} et d_{ji} par :

$$d_{ij} = \mathbf{M}_i^T\text{sign}(\mathbf{M}_j) \quad \text{et} \quad d_{ji} = \mathbf{M}_j^T\text{sign}(\mathbf{M}_i) \quad (10)$$

et on les normalise pour avoir $d_{ji}^2 + d_{ij}^2 = 1$ (même normalisation que $\frac{\mathbf{D}}{\|\mathbf{D}\|^2}$ dans SS-FAPI). Après, on met à jour la matrice $\mathbf{Q}(t) = \mathbf{Q}(t-1) - \mu\mathbf{Z}$ avec la matrice $\mathbf{Z}$ est donnée par :

$$\mathbf{Z} = [0, ..., d_{ji}\mathbf{Q}_j(t-1), 0, ..., d_{ij}\mathbf{Q}_i(t-1), ...0] \quad (11)$$

Finalement, on met à jour $\mathbf{W}(t)$ par :

$$\begin{aligned}\mathbf{W}(t) &= (\mathbf{W}_{PS}(t-1) + \mathbf{e}'\mathbf{g}^T)(\mathbf{Q}(t-1) - \mu\mathbf{Z}) \\ &= \mathbf{W}(t-1) + \mathbf{e}'\tilde{\mathbf{g}}^T - \mu\tilde{\mathbf{W}} \end{aligned} \quad (12)$$

où $\tilde{\mathbf{W}} = \mathbf{W}_{PS}(t)\mathbf{Z}$ est la matrice donnée par :

$$\tilde{\mathbf{W}} = [0, ..., d_{ji}\mathbf{M}_j, 0, ..., d_{ij}\mathbf{M}_i, ...0] \quad (13)$$

Les détails de l'algorithme sont résumés dans **Algorithme 3** et il est désigné par SS-FAPI2.

Algorithme 3 SS-FAPI2

1: $\mathbf{W}_{PS}(t) = \mathbf{W}_{PS}(t-1) + \mathbf{e}'\mathbf{g}^T$ dernière itération de FAPI
2: $\tilde{\mathbf{g}} = \mathbf{Q}(t-1)^T\mathbf{g}$
3: Choix des indices (i,j)
4: $\mathbf{M}_i = \mathbf{W}_i(t-1) + \tilde{g}_i\mathbf{e}'$ et $\mathbf{M}_j = \mathbf{W}_j(t-1) + \tilde{g}_j\mathbf{e}'$
5: $d_{ij} = \mathbf{M}_i^T\text{sign}(\mathbf{M}_j)$ et $d_{ji} = \mathbf{M}_j^T\text{sign}(\mathbf{M}_i)$
6: $d_{ij} = \frac{d_{ij}}{\sqrt{d_{ij}^2+d_{ji}^2}}$ et $d_{ji} = \frac{d_{ji}}{\sqrt{d_{ij}^2+d_{ji}^2}}$
7: $\mathbf{Z} = [0, ..., d_{ji}\mathbf{Q}_j(t-1), 0, ..., d_{ij}\mathbf{Q}_i(t-1), ...0]$
8: $\mathbf{Q}(t) = \mathbf{Q}(t-1) - \mu\mathbf{Z}$
9: $\tilde{\mathbf{W}} = [0, ..., d_{ji}\mathbf{M}_j, 0, ..., d_{ij}\mathbf{M}_i, ...0]$
10: $\mathbf{W}(t) = \mathbf{W}(t-1) + \mathbf{e}'\tilde{\mathbf{g}}^T - \mu\tilde{\mathbf{W}}$

3.4 Résultats de simulation et discussion

Nous présentons ici quelques simulations numériques pour évaluer les performances des algorithmes proposés et nous les comparons avec les algorithmes FAPI [6] et ℓ_1-PAST [8]. On considère le modèle de données $\mathbf{X} = \mathbf{AS} + \mathbf{N}$ avec $\mathbf{A} \in \mathbb{R}^{n\times p}$ une matrice de mélange parcimonieuse aléatoire (on utilise la fonction SPRANDN de MATLAB pour la générer), $\mathbf{S} \in \mathbb{R}^{p\times T}$ une matrice des signaux sources qui suit une distribution gaussienne de moyenne nulle et de variance unitée (T est le nombre des échantillons) et $\mathbf{N} \in \mathbb{R}^{n\times T}$ est un bruit gaussien blanc. Les indices de performances utilisés sont la norme ℓ_1 de la matrice de poids (avec les colonnes normalisées) pour la mesure de la parcimonie et l'erreur quadratique moyenne normalisée donnée par (14) pour la performance en sous-espace.

$$\rho(t) = \frac{1}{r}\sum_{i=1}^{r} \frac{trace(\mathbf{W}_i^{\#}(t)(\mathbf{I}_n - \mathbf{W}_{ex}\mathbf{W}_{ex}^T)\mathbf{W}_i(t))}{trace(\mathbf{W}_i^{\#}(t)\mathbf{W}_{ex}\mathbf{W}_{ex}^T\mathbf{W}_i(t))} \quad (14)$$

avec $r = 150$ le nombre de tirages de Monte-Carlo, $\mathbf{W}_i(t)$ est la matrice du sous-espace estimée au tirage i et à l'itération t (on note par $\mathbf{W}_i^{\#}(t)$ sa pseudo inverse), et $\mathbf{W}_{ex}$ est le sous-espace orthogonal exact calculé à partir de la matrice $\mathbf{A}$.

Figure 1 présente les performances de FAPI, ℓ_1-PAST et des algorithmes proposés SS-FAPI, SS-FAPI2 et SS-LORAF2. L'algorithme FAPI est utilisé comme un témoin d'une parcimonie faible. On constate que SS-FAPI et SS-FAPI2 présentent les meilleurs résultats avec une convergence plus lente de ce dernier en norme ℓ_1. On remarque que du point de vue sous-espace l'algorithme SS-LORAF2 est le meilleur mais il n'arrive pas à atteindre des niveaux faibles de la norme ℓ_1. L'algorithme ℓ_1-PAST présente la solution la moins bonne en sous-espace et

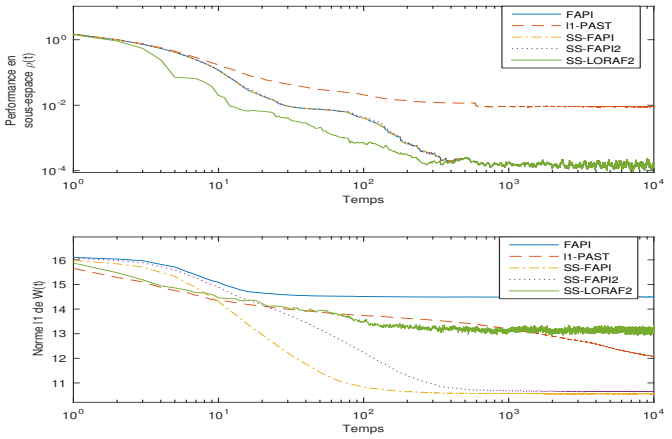

FIGURE 1 – L'erreur $\rho(t)$ et $\|\mathbf{W}\|_1$ vs temps avec : ($n = 16, p = 5, SNR = 25dB$)

avec une convergence en norme ℓ_1 très lente. On va discuter dans quelques points les remarques sur les simulations :

• Le choix de $\mu \geq 0$ est important, en effet si $\mu = 0$ on aura la même solution que FAPI (ou LORAF2), et si μ est "trop"grand on peut perdre la contrainte du rang de la solution ($\mathbf{Q}(t)$ devient proche de singulière pour une longue période d'exécution). Le fait de ne prendre que deux éléments de la matrice $\mathbf{D}$ dans SS-FAPI2 est judicieux pour éviter ce problème ; en effet pour que $\mathbf{Q}(t)$ soit inversible en supposant que $\mathbf{Q}(t-1)$ l'était, il suffit d'éviter les veleurs de μ qui vérifie $1 - \mu^2 d_{ij} d_{ji} = 0$ sachant que $d_{ij}^2 + d_{ji}^2 = 1$. On a pensé à choisir un μ adaptatif et on a constaté que si on prend $\mu_j = \sum_{i=1}^{n} |\mathbf{M}_{ij}|$ et que l'on change l'étape 4 de SS-FAPI par $\mathbf{Q}(t) = \mathbf{Q}(t-1)(\mathbf{I}_p - \mathbf{D}\text{diag}(\mu_1, \cdots, \mu_p))$, on améliore la convergence et la stabilité de SS-FAPI. Ceci est illustré par la figure 2 où l'on note cette variante SS-FAPI-adpt. Pour SS-FAPI2, nous avons proposé d'ajouter une simple étape de comparaison entre $\|\mathbf{W}(t)\|_1$ et $\|\mathbf{W}(t-1)\|_1$ qui décidera si l'on garde le μ (grand par défaut) ou si l'on doit prendre un μ plus faible par exemple $\frac{\mu}{5}$. On désigne cette variante par SS-FAPI2-adpt dans la figure 2.

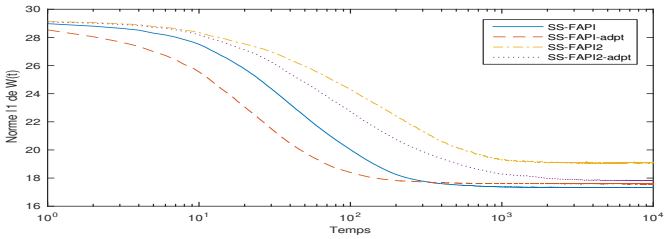

FIGURE 2 – $\|\mathbf{W}\|_1$ vs temps (n=16, p=9, SNR=25dB)

• Le choix de l'algorithme FAPI est basé sur sa bonne performance en poursuite sous-espace et surtout sur la propriété de variation lente de la solution, cette dernière est fondamentale pour le bon fonctionnement de notre algorithme à cause du gradient naturel utilisé (De manière générale, les algorithmes de type gradient sont mal adaptés aux variations rapides du système considéré). Ceci aussi explique l'échec de SS-LORAF2 d'atteindre des niveaux faibles en norme ℓ_1.

• La complexité de l'algorithme FAPI est $3np + O(p^2)$, ce qui fait que la complexité de SS-FAPI2 est égale à $4np + O(p^2)$ et elle est la plus faible en comparaison avec celle de SS-FAPI qui est égale à $2np^2 + O(p^3)$ ou celle de ℓ_1-PAST égale à $3np^2 + O(p^2)$. Cependant, le prix à payer pour cette réduction de coût est une convergence plus lente de l'algorithme SS-FAPI2 comparée à celle de SS-FAPI.

4 Conclusion

Nous avons élaboré de nouveaux algorithmes pour l'estimation adaptative du sous-espace principal sous la contraient de parcimonie de la matrice de poids. Ces algorithmes suivent une approche en deux étapes où ; dans la première on utilise l'algorithme FAPI pour extraire adaptativement une base orthonormale du sous-espace signal, et dans la deuxième on cherche la matrice de poids la plus parcimonieuse possible. Nous avons montré que l'on peut réduire la complexité de notre solution au prix d'une réduction de sa vitesse de convergence. De nombreuses applications sont possibles et peuvent être traitées par nos algorithmes qui présentent les avantages d'un faible coût de calcul et d'une performance améliorée par rapport à ℓ_1-PAST.

Références

[1] E. Oja, "Simplified neuron model as a principal component analyzer," *Journal of Mathematical Biology*, 1982.

[2] B. Yang, "Projection approximation subspace tracking," *IEEE Transactions on Signal Processing*, 1995.

[3] K. Abed-Meraim, A. Chkeif, and Y. Hua, "Fast orthonormal PAST algorithm," *IEEE Signal Processing Letters*, 2000.

[4] P. Strobach, "Low-rank adaptive filters," *IEEE Transactions on Signal Processing*, 1996.

[5] R. Badeau, G. Richard, B. David, and K. Abed-Meraim, "Approximated power iterations for fast subspace tracking," in *Seventh International Symposium on Signal Processing and Its Applications, 2003. Proceedings.*, 2003.

[6] R. Badeau, B. David, and G. Richard, "Fast approximated power iteration subspace tracking," *IEEE Transactions on Signal Processing*, 2005.

[7] K. Benidis, Y. Sun, P. Babu, and D. P. Palomar, "Orthogonal sparse PCA and covariance estimation via procrustes reformulation," *IEEE Transactions on Signal Processing*, 2016.

[8] X. Yang, Y. Sun, T. Zeng, T. Long, and T. K. Sarkar, "Fast STAP method based on PAST with sparse constraint for airborne phased array radar," *IEEE Transactions on Signal Processing*, 2016.