Algorithmes Adaptatifs Rapides Pour l'Estimation des Vecteurs Propres Mineurs

THAMERI MESSAOUD[1], ABED-MERAIM KARIM[1], BELOUCHRANI ADEL[2]

[1]Telecom-ParisTech, Département TSI
46 rue Barrault, 75634 Paris Cedex 13, France

[2]Ecole Polytechnique d'Alger
10 Avenue Hassen Badi, 16200 Alger, Algérie

thameri@telecom-paristech.fr, karim.abed@telecom-paristech.fr
adel.belouchrani@enp.edu.dz

Résumé – Dans cet article, nous proposons des algorithmes adaptatifs pour l'estimation des vecteurs propres mineurs d'une matrice de covariance empirique donnée. Dans le premier algorithme, nous proposons d'abord de modifier la matrice de covariance de manière à ce que les vecteurs propres mineurs deviennent des vecteurs propres principaux de la matrice transformée. Nous utilisons ensuite une version adaptée de l'algorithme GOPAST pour l'estimation des vecteurs propres désirés. Cet algorithme est de faible coût mais aussi de performances limitées pour des matrices de grandes tailles. Nous proposons alors un deuxième algorithme qui estime le sous espace mineur en tant que complément orthogonal du sous espace principal. Les vecteurs propres sont ensuite obtenus du sous espace mineur par des rotations de Givens. Cet algorithme est efficace en grande dimension mais sa complexité est élevée lorsque le rang du sous espace principal est grand. Pour pallier à cela, nous proposons une troisième méthode qui est une modification de MC-YAST permettant une réduction significative de la complexité de calcul par rapport à la méthode originale. Les algorithmes proposés sont dit 'rapides' dans le sens où leur complexité est linéaire. Les simulations numériques réalisées montrent aussi leurs efficacités dans différents scénarios pratiques.

Abstract – In this paper, we introduce new adaptive algorithms for the minor components estimation. At first, we propose to modify the covariance matrix in such a way the minor components become principal components of the transformed matrix. Then, an adapted version of the GOPAST algorithm is used to extract the desired vectors. This algorithm is shown to be inefficient for large size matrices and hence a second algorithm is proposed which estimates the desired minor components from the orthogonal complement of the principal subspace estimated first by OPAST. The limitation of the latter algorithm is its high computational cost when the principal subspace dimension is large. To alleviate this drawback, we introduce a third algorithm as a modified version of MC-YAST that preserves (and even improves) its efficiency while reducing significantly its cost. Along all this work, our main targets are the linear complexity and the efficiency for large dimensional systems.

1 Introduction

L'analyse en composantes principales (PCA : Principal Component Analysis) et en composantes mineurs (MCA : Minor Component Analysis) sont deux problèmes importants qui sont rencontrés dans différents domaines de traitement de l'information [4] pour la compression des données, pour l'estimation paramétriques (méthodes sous-espaces), pour l'optimisation quadratique, ou pour le filtrage rapide à rang réduit.
Dans cet article, on traite un problème relativement difficile qui est celui de l'estimation des vecteurs propres mineurs de matrices à grandes dimensions tout en maintenant une complexité linéaire $O(nm)$ où n est la taille du vecteur d'observation et m est le nombre de vecteurs propres mineurs à estimer.
Pour arriver à cette fin, on a proposé de nouvels algorithmes adaptatifs qui sont des versions modifiées et adaptées de certaines techniques PCA ou MCA utilisant la méthode de puissance [7]. Les caractéristiques principales de ces algorithmes sont leurs faibles complexités de calcul, leur relative efficacité en grandes dimensions et l'orthogonalité exacte des vecteurs propres estimés à chaque itération. Les algorithmes proposés sont dits 'rapides' dans le sens où leurs complexité est de l'ordre $O(nm)$ flops [1] par itération.

Considérons le vecteur d'entrée, de dimension n, $\mathbf{x}(t)$ $t = 1, 2, \cdots$, sa matrice de covariance associée $\mathbf{C}_{xx}$ est estimée comme suit

$$\mathbf{C}_{xx}(t) = \beta \mathbf{C}_{xx}(t-1) + (1-\beta)\mathbf{x}(t)\mathbf{x}^H(t) \qquad (1)$$

où β et un facteur d'oubli $0 < \beta < 1$.
La décomposition propre de cette matrice de covariance est

$$\mathbf{C}_{xx} = \mathbf{V}\mathbf{D}\mathbf{V}^H \qquad (2)$$

avec $\mathbf{D} = diag(d_1, \cdots, d_n)$ matrice diagonale contenant les valeurs propres en ordre décroissant (i.e. $d_1 > \cdots > d_n \geq 0$) et $\mathbf{V} = [\mathbf{v}_1, \cdots, \mathbf{v}_n]$ la matrice orthogonale des vecteurs propres associés. Nous nous intéressons ici à l'estimation adap-

1. Un flop correspond ici à une multiplication plus une addition.

tative des m vecteurs propres mineurs, i.e. $[\mathbf{v}_{n-m+1}, \cdots, \mathbf{v}_n]$ par des méthodes de complexités linéaires.

2 Algorithmes

Dans cette section, nous introduisons les algorithmes proposés pour la résolution du problème MCA. Comme nous manquons d'espace, nous avons choisi d'omettre certains détails techniques et d'insister principalement sur les démarches méthodologiques que nous avons adopté. Aussi, nous présentons les algorithmes dans un ordre qui respecte celui que nous avons suivi pour résoudre au fur et à mesure les problèmes rencontrés ou observés par simulations.

2.1 Algorithme 1 :

On part du constat que l'estimation PCA est beaucoup plus facile que l'estimation MCA et l'on remplace la matrice de covariance $\mathbf{C}_{xx}$ par la matrice $\mathbf{C}_{xx}^{'}$ définie par

$$\mathbf{C}_{xx}^{'} = \alpha\mathbf{I} - \mathbf{C}_{xx} = \mathbf{V}\left[\alpha\mathbf{I} - \mathbf{D}\right]\mathbf{V}^H \tag{3}$$

$\mathbf{C}_{xx}^{'}$ et $\mathbf{C}_{xx}$ ont la même base de vecteurs propres à la différence que, par cette transformation, les vecteurs propres mineurs de $\mathbf{C}_{xx}$ deviennent les vecteurs propres principaux de $\mathbf{C}_{xx}^{'}$. Cette dernière est estimée en régime adaptatif, par la récurrence suivante :

$$\mathbf{C}_{xx}^{'}(t) = \beta\mathbf{C}_{xx}^{'}(t-1)+(1-\beta)(\mathbf{b}(t)\mathbf{b}^H(t)-\mathbf{x}(t)\mathbf{x}^H(t)) \tag{4}$$

où $\mathbf{b}(t)$ est un processus aléatoire choisi satisfaisant $\mathbf{C}_{bb} = E(\mathbf{b}(t)\mathbf{b}^H(t)) = \alpha\mathbf{I}$. L'extraction des m vecteurs propres principaux de $\mathbf{C}_{xx}^{'}$ est réalisée par une version adaptée de l'algorithme H-GOPAST [1]. Ce dernier est une implémentation rapide de la méthode de puissance où la matrice $\mathbf{W}(t)$ générant le sous espace principal est calculée itérativement par :

$$\mathbf{C}_{xy}^{'}(t) = \mathbf{C}_{xx}^{'}(t)\mathbf{W}(t-1) \tag{5}$$

$$\mathbf{W}(t)\mathbf{R}(t) = \mathbf{C}_{xy}^{'}(t) \tag{6}$$

où $\mathbf{R}(t)$ est la racine carrée hermitienne de $\mathbf{C}_{xy}^{'}(t)^H\mathbf{C}_{xy}^{'}(t)$. Une mise à jour rapide des équations précédentes basée sur la technique d'approximation de projection [2], i.e. $\mathbf{C}_{xx}^{'}(t)\mathbf{W}(t) \approx \mathbf{C}_{xx}^{'}(t)\mathbf{W}(t-1)$ permet d'atteindre une complexité de calcul linéaire de l'ordre de $O(nm)$. La matrice $\mathbf{W}$ génère le sous espace principal de $\mathbf{C}_{xx}^{'}$, i.e. $\mathbf{W} = \mathbf{U}_s\mathbf{Q}$, où $\mathbf{U}_s$ est la matrice des vecteurs propres principaux de $\mathbf{C}_{xx}^{'}$ (vecteurs propres mineurs de $\mathbf{C}_{xx}$) et $\mathbf{Q}$ une matrice unitaire donnée. La matrice $\mathbf{Q}$ restante est identifiée par diagonalisation de la matrice [2] $\mathbf{Z} = (\mathbf{W}^H\mathbf{C}_{xx}^{'}\mathbf{W})^{-1}$ sous forme de produit de rotations de Givens successives (voir détails dans [8] et ses références).

Soulignons que dans cette version adaptée, on utilise (4) au lieu de (1) (comme dans la version originale), ce qui conduit à une légère augmentation de la complexité de calcul qui reste toutefois d'ordre linéaire, $O(nm)$.

Aussi, le choix du paramètre α est très important pour les bonnes performances de l'algorithme : α doit être suffisamment grand pour garantir le caractère définie positive de $\mathbf{C}_{xx}^{'}$ mais pas trop grand pour que les valeurs propres de $\mathbf{C}_{xx}^{'}$ ne soient pas très proches (ce qui rend le problème très difficile même pour l'extraction des vecteurs propres principaux). Une solution simple considérée ici consiste à choisir $\alpha = S = trace(\mathbf{C}_{xx})$ que l'on estime adaptativement par :

$$S(t) = \beta S(t-1) + (1-\beta)\left\|\mathbf{x}(t)\right\|^2. \tag{7}$$

2.2 Algorithme 2 :

Pour compenser les insuffisances de l'algorithme précédent lorsque n et m sont grands (voir résultats de simulation), nous proposons un second algorithme qui estime le sous espace mineur comme complément orthogonal du sous espace principal. En particulier, dans le cas où $p = n - m \ll n$, la PSA (Principal Subspace Analysis) est peu coûteuse et très précise ce qui conduit à de très bonnes performances (en termes de coût et de précision d'estimation) de l'algorithme considéré. Dans ce travail, l'analyse PSA est réalisée par l'algorithme OPAST [6]. Nous décrivons brièvement ci-dessous comment passer de la PSA à la MSA (Minor Subspace Analysis) ensuite à la MCA en maintenant une complexité de calcul linéaire.

Calcul rapide du complément orthogonal (PSA à MSA) : Soit $\mathbf{W}$ la matrice $n \times (n-m)$ générant le sous espace principal de $\mathbf{C}_{xx}$. Pour calculer un complément orthogonal, décomposons $\mathbf{W}$ sous la forme :

$$\mathbf{W} = \begin{bmatrix}\mathbf{W}_1 \\ \mathbf{W}_2\end{bmatrix}$$

où $\mathbf{W}_1$ est de taille $m \times (n-m)$ et $\mathbf{W}_2$ est de taille $(n-m) \times (n-m)$. On peut considérer que les $(n-m)$ lignes de $\mathbf{W}_2$ constituent une base ce qui permet d'écrire les lignes de $\mathbf{W}_1$ en fonction de celles de $\mathbf{W}_2$

$$\mathbf{W}_1 = \mathbf{P}^H\mathbf{W}_2 \tag{8}$$

où $\mathbf{P}$, de taille $(n-m) \times m$, est la matrice de passage calculée comme suit

$$\mathbf{P}^H = \mathbf{W}_1\mathbf{W}_2^{-1} \tag{9}$$

Nous construisons alors une base du complément orthogonal (i.e. sous espace mineur) sous la forme de la matrice $\mathbf{U}$ de taille $n \times m$ définie par $\mathbf{U} = \begin{bmatrix}\mathbf{I}_m \\ -\mathbf{P}\end{bmatrix}$ et qui vérifie bien

$$\mathbf{U}^H\mathbf{W} = \mathbf{W}_1 - \mathbf{P}^H\mathbf{W}_2 = \mathbf{0}.$$

Dans notre implémentation adaptative, le sous espace principal est mis à jour par l'algorithme OPAST sous la forme $\mathbf{W}(t) = \mathbf{W}(t-1) + \mathbf{e}\mathbf{q}^H$ où $\mathbf{e}$ et $\mathbf{q}$ sont des vecteurs de tailles respectives $n \times 1$ et $m \times 1$ donnés dans [6]. On peut donc écrire

$$\mathbf{W}_1(t) = \mathbf{W}_1(t-1) + \mathbf{e}_1\mathbf{q}^H \tag{10}$$

$$\mathbf{W}_2(t) = \mathbf{W}_2(t-1) + \mathbf{e}_2\mathbf{q}^H \tag{11}$$

2. Cette matrice est déjà calculée dans l'étape précédente pour l'estimation de $\mathbf{W}(t)$.

En utilisant le lemme de Schur pour l'inversion matricielle [3], nous obtenons

$$\mathbf{W}_2^{-1}(t) = [\mathbf{W}_2(t-1) + \mathbf{e}_2\mathbf{q}^H]^{-1} = \mathbf{W}_2^{-1}(t-1) + \alpha \mathbf{E}_e \mathbf{E}_q^H \tag{12}$$

avec $\mathbf{E}_e = \mathbf{W}_2^{-1}(t-1)\mathbf{e}_2$, $\mathbf{E}_q = \mathbf{W}_2^{-1}(t-1)\mathbf{q}$ et $\alpha = \frac{-1}{1+\mathbf{q}^H\mathbf{E}_e}$, et finalement

$$\mathbf{P}^H(t) = \mathbf{P}^H(t-1) + \alpha \mathbf{W}_1(t-1)\mathbf{E}_e\mathbf{E}_q^H + \mathbf{e}_1\mathbf{q}^H\mathbf{W}_2^{-1}(t) \tag{13}$$

Le coût de ce calcul est de l'ordre de $O(pn)$ flops par itération (avec $p = n - m$).

Finalement, la matrice $\mathbf{U}$ ainsi calculée ne formant pas une base orthonormale du sous espace mineur, nous proposons de l'orthonormaliser par la transformation :

$$\mathbf{U} := \mathbf{U}(\mathbf{U}^H\mathbf{U})^{-\frac{1}{2}} \tag{14}$$

où $(\mathbf{U}^H\mathbf{U})^{-\frac{1}{2}}$ représente l'inverse de la racine carrée du produit $(\mathbf{U}^H\mathbf{U})$. Ce calcul est réalisé en $O(mp)$ flops grâce à la procédure d'orthonormalisation rapide proposée dans [7] dont on omet les détails ici.

Calcul rapide des vecteurs propres (MSA à MCA) : Soit $\mathbf{y}(t) = \mathbf{U}^H(t)\mathbf{x}(t)$. On peut observer que la pseudo-matrice de covariance

$$\mathbf{C}_{yy} = E\left[\mathbf{y}(t)\mathbf{y}^H(t)\right] = \mathbf{U}^H\mathbf{C}_{xx}\mathbf{U} \tag{15}$$

converge vers un matrice diagonale avec les m plus petites valeurs propres de $\mathbf{C}_{xx}$ quand $\mathbf{U}$ converge vers les vecteurs propres mineurs de cette dernière.

Dans notre schéma adaptatif, $\mathbf{C}_{yy}$ est évaluée par l'equation de récurrence suivante

$$\mathbf{C}_{yy}(t) = \beta\mathbf{C}_{yy}(t-1) + (1-\beta)\mathbf{y}(t)\mathbf{y}^H(t) \tag{16}$$

et sa diagonalisation est réalisée par l'utilisation de rotations de Givens (2 rotations par itération comme dans l'algorithme H-GOPAST [1]). La complexité numérique de ce dernier calcul est de l'ordre de $O(nm)$ flops par itération.

2.3 Algorithme 3 :

L'algorithme 2 est efficace quand $n - m \ll n$ auquel cas sa complexité est quasi linéaire [4]. Toutefois, on a beaucoup de situations où m est relativement petit (voire $m = 1$). Dans ce cas nous avons proposé de revenir sur un algorithme efficace qui est YAST [3] et de lui apporter des modifications pour réduire sa complexité de $O(n^2)$ à $O(nm)$ en utilisant la technique d'approximation de projection.

Les auteurs de [5] ont utilisé la technique du gradient conjugué pour améliorer les performance de YAST [3]. Leur idée est de remplacer le vecteur auxiliaire $\mathbf{x}(t)$ de la version originale par une estimée d'un vecteur propre mineur évalué par un gradient conjugué. Dans cette version améliorée de YAST, la complexité quadratique provient du produit de $\mathbf{C}_{xx}(t)$ par un vecteur de gradient $\mathbf{g}(t)$ (voir Table 1 de [5]). A la convergence, ce vecteur de gradient tend vers une valeur limite (fixe), ce qui permet de supposer sa variation faible entre deux itérations successives et d'utiliser ainsi l'approximation de projection qui permet de réduire le coût de calcul de $O(n^2)$ à $O(nm)$. Autrement dit, si on note $\mathbf{c}(t) = \mathbf{C}_{xx}(t)\mathbf{g}(t)$, alors grâce à l'approximation de projection, on peut écrire :

$$\mathbf{c}(t) \approx \beta\mathbf{c}(t-1) + (1-\beta)\mathbf{x}(t)(\mathbf{x}^H(t)\mathbf{g}(t)).$$

A l'exception de cette modification, le reste de l'algorithme est identique à celui de [5]. De manière surprenante, nous avons observé par simulation que la modification apportée permet aussi d'améliorer la précision d'estimation de cet algorithme.

3 Simulation

Dans cette section, on présente trois cas de figure qui illustrent bien les performances des algorithmes proposés (n petit, n grand et $n - m << n$, n grand et $m << n$).
Pour la comparaison des vecteurs propres mineurs, on utilise l'erreur quadratique moyenne (EQM) évaluée à l'instant i par :

$$\rho(i) = \frac{1}{mp_0}\sum_{p=1}^{p_0}\|\mathbf{U}^p(i) - \mathbf{U}^{ex}\|^2 \tag{17}$$

où $p_0 = 100$ est le nombre d'expériences, $\mathbf{U}^p(i)$ est la matrice des vecteurs propres mineurs [5] estimés à l'instant i de l'expérience p et $\mathbf{U}^{ex}$ est celle des vecteurs propres mineurs exacts. Nous ne montrons pas ici les mesures d'orthonormalité des matrices estimées car elles ne sont pas descriminatoires vu que toutes les méthodes considérées garantissent l'orthonormalité exacte de la matrice des vecteurs propres estimés à chaque itération. Dans nos simulations, le signal $\mathbf{x}(t)$ est généré aléatoirement selon une loi gaussienne de covariance $\mathbf{C}_{xx}$ définie positive.
La figure 1 illustre le cas où $n = 4$ (i.e. n faible) et $m = 2$ et où l'on compare les performances obtenues par l'algorithme 1 et MC-YAST dans [8]. On observe que dans ce cas de figure, l'algorithme proposé a des performances proches de celle de MC-YAST dont la complexité est d'ordre $O(n^2)$.
La figure 2 illustre le cas où $n = 20$ et $m = 15$ (n grand et $n - m$ faible) et montre que les performances de l'algorithme 1 se dégradent fortement quand la taille du système augmente alors que l'algorithme 2 présente des performances très bonnes comparées à celles de MC-YAST et de l'algorithme 1.
La figure 3 montre que la modification considérée (approximation de projection) a apporté ses fruits en terme de coût mais aussi par la réduction de l'erreur d'estimation. Une explication plausible de cette observation serait que l'approximation de projection permet plus de fluctuations du vecteur considéré

3. Le calcul numérique de l'inverse de $\mathbf{W}_2(t)$ peut s'avérer instable et donc, en pratique, on remplace cette dernière par $\mathbf{W}_2(t) + \epsilon I$ avec ϵ fixe et de faible valeur.

4. En fait, la complexité globale de ce second algorithme est quadratique en n car $p + m = n$ et donc $O(nm) + O(np) = O(n^2)$.

5. Pour cette comparaison, les vecteurs propres sont normalisés à l'unité et leurs phases sont choisies telles que le premier élement de chaque vecteur est réel positif.

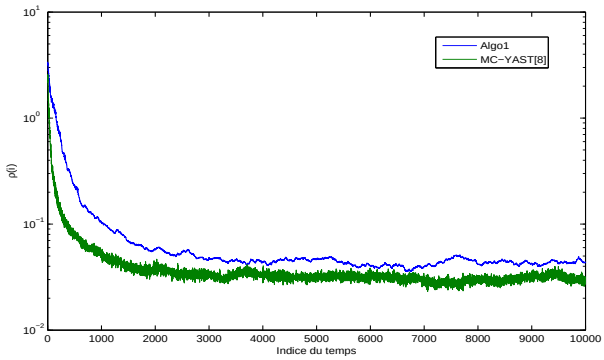

FIGURE 1 – EQM v.s. itération : $n = 4$ et $m = 2$.

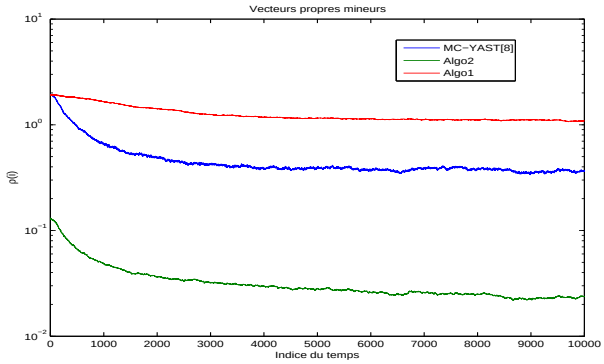

FIGURE 2 – EQM v.s. itération : $n = 20$ et $m = 15$.

et donc un meilleur balayage des directions de recherche du sous espace mineur.
Dans la figure 4 où $n = 20$ et $m = 10$, on illustre une situation intermédiaire qui permet d'observer l'avantage (en terme d'EQM) de l'algorithme 2 proposé mais pour une complexité qui est ici proche de $O(n^2)$.

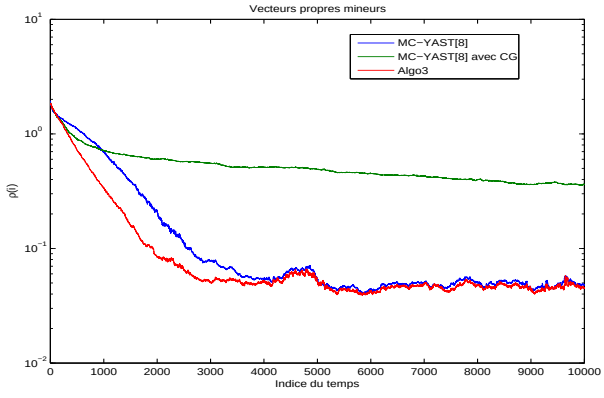

FIGURE 3 – EQM v.s. itération : $n = 20$ et $m = 5$.

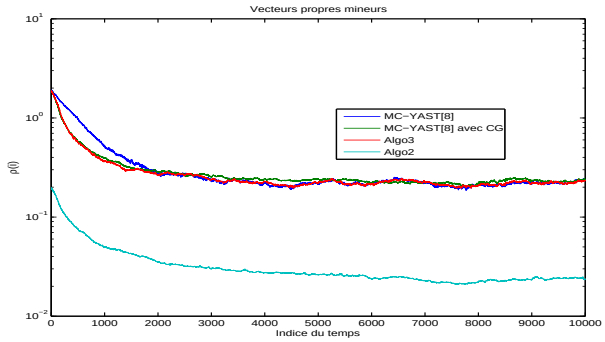

FIGURE 4 – EQM v.s. itération : $n = 20$ et $m = 10$.

4 Conclusion

Dans cet article, nous avons proposé des algorithmes adaptatifs rapides pour l'estimation des vecteurs propres mineurs. Le premier algorithme introduit est peu compétitif car inefficace pour des systèmes à grandes dimensions alors que les deux autres algorithmes considérés sont relativement efficaces dans ce contexte. L'avantage principal de l'algorithme 2 est sa bonne performance d'estimation (faible EQM) que nous avons observé sur tous les scénarios de simulation réalisés, alors que pour l'algorithme 3 l'avantage réside dans son faible coût de calcul (complexité linéaire dans tous les cas de figure).

Références

[1] M. Thameri, K Abed-Meraim et A. Belouchrani. *Fast PCA and Data Whitening Algorithms*. Proceedings WoSSPA 2011.

[2] Bin Yang, *Projection Approximation Subspace Tracking*, IEEE Tr. on Signal Processing, vol. 43, pp. 95–107, 1995.

[3] B. Roland , D. Bertrand et R. Gaël . *YAST Algorithm for Minor Subspace Tracking*. ICASSP 2006.

[4] Comon, P. ; Golub, G.H. ; *Tracking a few extreme singular values and vectors in signal processing*, Proceedings of the IEEE Volume : 78 , Issue : 8, pp : 1327 - 1343, 1990.

[5] Roland BADEAU, Bertrand DAVID, Gaël RICHARD, *Conjugate Gradient Algorithms for Minor Subspace Analysis*. ICASSP 2007.

[6] K. Abed-Meraim, A. Chkeif, Y. Hua, and S. Attallah, *Fast Orthonormal PAST Algorithm*, IEEE Signal Processing Letters. March 2000.

[7] K. Abed-Meraim, A. Chkeif, Y. Hua, and S. Attallah, *On a class of orthonormal algorithms for principal and minor subspace tracking*, Journal of VLSI Signal Processing Systems (invited paper), 2001.

[8] S. Bartelmaos et K. Abed-Meraim. *Fast minor component extraction using Givens rotations*, Electronics Letters, Volume : 43 , Issue : 18, pp : 1001 - 1003, 2007.