Fast exact variable order affine projection algorithm

Miguel Ferrer*, Alberto Gonzalez, Maria de Diego, Gema Piñero

Audio and Communications Signal Processing Group (GTAC)
Institute of Telecommunications and Multimedia Applications (iTEAM)
Universitat Politècnica de València

Abstract

Variable order affine projection algorithms have been recently presented to be used when not only the convergence speed of the algorithm has to be adjusted but also its computational cost and its final residual error. These kind of affine projection (AP) algorithms improve the standard AP algorithm performance at steady state by reducing the residual mean square error. Furthermore these algorithms optimize computational cost by dynamically adjusting their projection order to convergence speed requirements.

The main cost of the standard AP algorithm is due to the matrix inversion that appears in the coefficient update equation. Most efforts to decrease the computational cost of these algorithms have focused on the optimization of this matrix inversion. This paper deals with optimization of the computational cost of variable order AP algorithms by recursive calculation of the inverse signal matrix. Thus, a fast exact variable order AP algorithm is proposed. Exact iterative expressions to calculate the inverse matrix when the algorithm projection order either increases or decreases are incorporated into

☆Partially supported by TEC2009-13741, PROMETEO 2009/0013, GV/2010/027, ACOMP/2010/006 and UPV PAID-06-09.

*e-mail: mferrer@dcom.upv.es, Tel.: +34-96-3877007-88272, Fax: +34-96-3877309

a variable order AP algorithm leading to a reduced complexity implementation. The simulation results show the proposed algorithm performs similarly to the variable order AP algorithms and it has a lower computational complexity.

Keywords: Adaptive filters, Affine projection algorithm, Fast algorithm, Computational complexity, Efficient matrix inversion.

---

1. Introduction

The affine projection (AP) algorithm [1][2] shows better convergence speed than the least mean square (LMS) algorithm [3] and it is simple, robust and stable. The efficiency of AP algorithms has been reported in a variety of applications, such as active noise control [4], acoustic equalization [5] and echo cancellation [6]. The behavior of the AP is mainly determined by a parameter called projection order, N. The AP algorithm behaves similarly to the normalized LMS algorithm [3] when $N = 1$ and to the recursive least squares (RLS) adaptive algorithm [7] when N increases. Therefore the AP shows slow convergence and little residual error when N is small, and fast convergence and higher residual error for large values of N. Variable step-size affine projection algorithms have already been proposed [8]-[11] to overcome this duality and achieve better performance in steady state without penalizing the speed of adaptation of the algorithm. Although these strategies achieve better final error in steady state, their computational cost remains invariant throughout algorithm execution and depends mainly on its projection order. A possible improvement to overcome these drawbacks is the adaptation of the projection order in response to algorithm performance.

Thus, some variable order AP algorithms [12]-[14], have been developed recently in order to dynamically adjust their projection order to convergence speed needs, and decrease the computational cost of the algorithm and its residual error. However this promising improvement of the AP algorithm has still some performance points to be analyzed such as its computational cost, which involves developing its fast versions.

Despite its computational cost, the AP algorithm can be considered good enough [15] in comparison with other algorithms that exhibit similar performance like the RLS algorithm. Many efforts have been made to decrease its computational cost as described in [16]-[21]. However the strategies based on approximations or models used to decrease the computational cost of the algorithm can slightly worsen the algorithm performance in some cases. Therefore, this paper avoids these methods and focuses on the efficient calculation of the inverse signal matrix that appears within the algorithm update equations of all the variable order AP algorithms. By using this method, efficient approaches are obtained that behave exactly like the original non fast versions when an accurate initial value of the inverse matrix is provided and show a significant reduction of their computational cost. Variable order AP algorithms can eventually change their projection order between iterations, therefore a recursive method to calculate the inverse matrix has to consider the inverse matrix updates from a previous inverse matrix of different size. Among the different variable order AP algorithms available and to illustrate the performance of the efficient method introduced, the authors have used the variable order AP (VAP) algorithm. The application of the fast recursive method to the VAP provides the fast exact variable order AP (FExVAP).

On the other hand, the efficient computation of the matrix inversion can be applied to other variable order AP algorithms such as the evolving order AP (E-AP) algorithm [13]. Thus, some simulation results of the E-AP and of its fast exact approach, the FExE-AP, which uses the proposed fast exact inversion method, have been also carried out.

Section 2 briefly describes the AP algorithm and the foundation of its variable order versions, and a recursive method to calculate the inverse signal matrix from its previous values for the VAP algorithm is developed in Section 3. The simulation results are presented in Section 4, comparing the VAP, the E-AP, their fast exact approaches (FExVAP and FExE-AP, respectively) and the original AP algorithm. The reduction of the computational cost in terms of number of multiplications is also presented in Section 4. Finally conclusions are summarized in Section 5.

2. The variable order affine projection algorithm

The AP algorithm attempts to generate a version of an unknown signal, $d(n)$, by filtering a reference signal, $x(n)$, correlated with $d(n)$. Fig. 1 illustrates an example of system identification where $x(n)$ and $d(n)$ are related through a transversal adaptive filter. The adaptive filter coefficients are updated by the following equation [22] for a projection order N,

$$\mathbf{w}_L(n) = \mathbf{w}_L(n-1) + \mu \mathbf{A}^T(n)[\mathbf{A}(n)\mathbf{A}^T(n) + \delta \mathbf{I}]^{-1}\mathbf{e}_N(n) \tag{1}$$

where $\mathbf{I}$ represents the $N \times N$ identity matrix, $\mathbf{w}_L(n)$ is a vector that comprises the L adaptive filter coefficients and matrix $\mathbf{A}(n)$ of $N \times L$ size is defined as

$$\mathbf{A}^T(n) = [\mathbf{x}_L(n)\ \mathbf{x}_L(n-1)\ ...\ \mathbf{x}_L(n-N+1)], \tag{2}$$

with $\mathbf{x}_L^T(n) = [x(n)\ x(n-1)\ ...\ x(n-L+1)]$ and $\mathbf{e}_N(n)$ is given by

$$\mathbf{e}_N(n) = \mathbf{d}_N(n) - \mathbf{A}(n)\mathbf{w}_L(n-1), \tag{3}$$

with

$$\mathbf{d}_N^T(n) = [d(n)\ d(n-1)\ ...\ d(n-N+1)]. \tag{4}$$

Constants μ and δ are called, respectively, the convergence and the regularization parameters [22].

Variable order AP algorithms use the update equation (1) but their projection order can change between iterations. This projection order varies in order to speed up convergence speed and minimize computational cost and residual error depending on certain conditions that can differ slightly between different variable order AP approaches. For instance, the evolving order AP described in [13] uses the instantaneous value of the residual error signal power to update the projection order and keep a single μ. Even though this lead to improvement due to the change in projection order, as a general rule, it does not achieve optimum residual error in the steady state since this residual error depends on both μ and the projection order. Alternatively, the AP approach used in this paper and named variable order AP (VAP), changes the projection order when the, also variable, convergence parameter μ exceeds given maximum or minimum values. Therefore, the algorithm described (VAP) changes both the step-size parameter and the projection order (a similar AP algorithm for echo cancellation that changes also both parameters is presented in [23]). Thus,

$$N(n+1) = \begin{cases} \min\{\mathrm{N(n)}+1, \mathrm{N}_{\max}\}, & \mu(n) > \mu_{\max}\cdot\mu_{N\mathrm{up}} \\ N(n), & \text{other} \\ \max\{\mathrm{N(n)}-1, 1\}, & \mu(n) < \mu_{\max}\cdot\mu_{N\mathrm{down}}, \end{cases} \tag{5}$$

where $N_{\max}$ is the higher projection order and $\mu_{N\mathrm{up}}$ and $\mu_{N\mathrm{down}}$ the maximum and minimum thresholds relative to $\mu_{\max}$. Due to the simplicity of the algorithm, these parameters are selected depending on the objectives. In this way, if the goal is to perform well at transient state we need a low μ_{Ndown} value, whereas a low μ_{Nup} value provides good tracking capabilities. On the other hand, high threshold values are required to obtain a good steady-state behavior. The maximum step-size parameter $\mu_{\max}$ in (5) is chosen to guarantee both fast convergence speed and filter stability and ideally should be less than 1 [24][25].

The variation rule for the convergence parameter can be chosen by attempting to ensure that the mean square deviation of the filter weights undergoes the largest decrease between algorithm iterations and it is given by [8]

$$\mu(n) = \mu_{\max}\frac{\|\mathbf{p}(n)\|^2}{\|\mathbf{p}(n)\|^2 + C}, \tag{6}$$

where $\mathbf{p}(n)$ is an estimation of the mean value of $\mathbf{A}^T(n)[\mathbf{A}(n)\mathbf{A}^T(n)+\delta\mathbf{I}]^{-1}\mathbf{e}_N(n)$, which is obtained from an exponential weighting of its instantaneous value as

$$\mathbf{p}(n) = \alpha\mathbf{p}(n-1) + (1-\alpha)\mathbf{A}^T(n)[\mathbf{A}(n)\mathbf{A}^T(n)+\delta\mathbf{I}]^{-1}\mathbf{e}_N(n) \tag{7}$$

with $0 < \alpha < 1$, and C is a positive parameter that depends on the algorithm projection order. $\mu(n)$ is equivalent to the constant μ parameter in (1).

Moreover, inversion of the $\mathbf{R}_N(n)$ matrix, being $\mathbf{R}_N(n) = \mathbf{A}(n)\mathbf{A}^T(n) + \delta\mathbf{I}$, requires $O(N^3/2)$ multiplications, which can represent the costlier part of the algorithm. The size of this matrix changes dynamically in variable order AP algorithms. There are methods to recursively calculate this matrix inversion with a much lower cost but they have to be extended to the variable order approach, which means considering cases when the projection order either increases or decreases between algorithm iterations.

3. Efficient matrix inversion

As noted above, the costlier computational cost of the AP algorithm is initially due to the computation of the matrix $\mathbf{R}_N(n)$ and its inversion, which is given by $LN^2 + O(N^3/2)$ multiplications. However, $\mathbf{A}(n)\mathbf{A}^T(n)$ can be computed recursively as

$$\mathbf{A}(n)\mathbf{A}^T(n) = \mathbf{M}(n) = \mathbf{M}(n-1) + \mathbf{x}_N(n)\mathbf{x}_N^T(n) - \mathbf{x}_N(n-L)\mathbf{x}_N^T(n-L), \quad (8)$$

using $2N^2$ multiplications, thus this cost is reduced to $2N^2 + O(N^3/2)$ multiplications. In order to further reduce this computational cost, recursive algorithms to calculate $\mathbf{R}_N^{-1}(n)$ from the matrix in the previous iterations, $\mathbf{R}_N^{-1}(n-1)$, can be used. Nevertheless, recursive algorithms to calculate either $\mathbf{R}_{N-1}^{-1}(n)$ or $\mathbf{R}_{N+1}^{-1}(n)$ from the previous values with different projection order, that is from $\mathbf{R}_N^{-1}(n-1)$, also have to be developed in order to deal with the recent variable order versions of the AP algorithm. In **Algorithm 1** the different cases discussed below for the FExVAP algorithm are summarized.

3.1. Iterative calculation of $\mathbf{R}_N^{-1}(n)$ *from* $\mathbf{R}_N^{-1}(n-1)$

The iterative calculation of $\mathbf{R}_N^{-1}(n)$ from $\mathbf{R}_N^{-1}(n-1)$ is similar to the matrix inversion used in the sliding window RLS algorithm [7]. An equivalent method can be found in [26] or [27].

From (8) the following equations can be given

$$\begin{aligned}&\mathbf{R}_N(n) = \mathbf{R}_N(n-1) + \mathbf{x}_N(n)\mathbf{x}_N^T(n) - \mathbf{x}_N(n-L)\mathbf{x}_N^T(n-L)\\ &= \mathbf{Q}_N(n) - \mathbf{x}_N(n-L)\mathbf{x}_N^T(n-L)\end{aligned} \tag{9}$$

with

$$\mathbf{Q}_N(n) = \mathbf{R}_N(n-1) + \mathbf{x}_N(n)\mathbf{x}_N^T(n), \tag{10}$$

and $\mathbf{x}_N^T(n) = [x(n)\ x(n-1)\ ...\ x(n-N+1)]$.

We can carry out the following matrix identification in (9)

1. $\mathbf{\Gamma} = \mathbf{R}_N(n)$
2. $\mathbf{\Theta}^{-1} = \mathbf{Q}_N(n)$
3. $\mathbf{\Phi} = \mathbf{x}_N(n)$
4. $\mathbf{\Psi} = -1$

and then apply the matrix inversion lemma (see Appendix A) to calculate $\mathbf{R}_N^{-1}(n)$ as

$$\mathbf{R}_N^{-1}(n) = \mathbf{Q}_N^{-1}(n) + \boldsymbol{\beta}(n)[1 - \mathbf{x}_N^T(n-L)\boldsymbol{\beta}(n)]^{-1}\boldsymbol{\beta}^T(n), \tag{11}$$

where $\boldsymbol{\beta}(n) = \mathbf{Q}_N^{-1}(n)\mathbf{x}_N(n-L)$.

The matrix inversion lemma can be used again to calculate $\mathbf{Q}_N^{-1}(n)$, using in (10)

1. $\mathbf{\Gamma} = \mathbf{Q}_N(n)$

2. $\mathbf{\Theta}^{-1} = \mathbf{R}_N(n-1)$
3. $\mathbf{\Phi} = \mathbf{x}_N(n)$
4. $\mathbf{\Psi} = 1$

Thus

$$\mathbf{Q}_N^{-1}(n) = \mathbf{R}_N^{-1}(n-1) - \boldsymbol{\alpha}_1(n)[1 + \mathbf{x}_N^T(n)\boldsymbol{\alpha}_1(n)]^{-1}\boldsymbol{\alpha}_1^T(n) \qquad (12)$$

where $\boldsymbol{\alpha}_1(n) = \mathbf{R}_N^{-1}(n-1)\mathbf{x}_N(n)$. Therefore, a recursive algorithm to calculate $\mathbf{R}_N^{-1}(n)$ from $\mathbf{R}_N^{-1}(n-1)$ is finally summarized in **Algorithm 1**.

In this way $4(N^2+N)$ multiplications are needed to obtain $\mathbf{R}_N^{-1}(n)$ using the above fast exact strategy, thereby reducing the original cost mainly for high projection orders.

3.2. Iterative calculation of $\mathbf{R}_{N-1}^{-1}(n)$ *from* $\mathbf{R}_N^{-1}(n-1)$

This calculation is made in two steps. First $\mathbf{R}_N^{-1}(n)$ from $\mathbf{R}_N^{-1}(n-1)$ is calculated by using the matrix inversion lemma described in Section 3.1. Then, a simple relation allow to calculate $\mathbf{R}_{N-1}^{-1}(n)$ from $\mathbf{R}_N^{-1}(n)$. To achieve this, matrix $\mathbf{R}_N(n)$ can be rewritten as

$$\mathbf{R}_N(n) = \begin{pmatrix} \mathbf{R}_{N-1}(n) & \mathbf{r}_{N-1}(n) \\ & \\ \mathbf{r}_{N-1}^T(n) & r_{N-1}(n) \end{pmatrix}, \qquad (13)$$

where $\mathbf{r}_{N-1}(n)$ and $r_{N-1}(n)$ can be also recursively calculated by

$$\mathbf{r}_{N-1}(n) = \mathbf{r}_{N-1}(n-1) + \mathbf{x}_{N-1}(n)x(n-N+1) - \mathbf{x}_{N-1}(n-L)x(n-N-L+1), \qquad (14)$$

and

$$r_{N-1}(n) = r_{N-1}(n-1) + x^2(n-N+1) - x^2(n-N-L+1). \qquad (15)$$

From (B.1) in Appendix B and by using the following identification statements,

1. $\mathbf{A}_{11} = \mathbf{R}_{N-1}(n)$
2. $\mathbf{A}_{12} = \mathbf{r}_{N-1}(n)$
3. $\mathbf{A}_{21} = \mathbf{r}^T_{N-1}(n)$
4. $\mathbf{A}_{22} = r_{N-1}(n)$
5. $\mathbf{F}^{-1}_{11} = (\overline{\mathbf{R}})^{-1}_N(n)$, which comprises $N-1 \times N-1$ upper left elements of $\mathbf{R}^{-1}_N(n)$,

we can rewrite the inverse of (13) as

$$\mathbf{R}^{-1}_N(n) = \begin{pmatrix} (\overline{\mathbf{R}})^{-1}_N(n) & -(\overline{\mathbf{R}})^{-1}_N(n)\mathbf{r}_{N-1}(n)r^{-1}_{N-1}(n) \\ \\ -r^{-1}_{N-1}(n)\mathbf{r}^T_{N-1}(n)(\overline{\mathbf{R}})^{-1}_N(n) & r^{-1}_{N-1}(n) + r^{-1}_{N-1}(n)\mathbf{r}^T_{N-1}(n)(\overline{\mathbf{R}})^{-1}_N(n)\mathbf{r}_{N-1}(n)r^{-1}_{N-1}(n) \end{pmatrix}. \tag{16}$$

Note that $\mathbf{R}^{-1}_N(n)$ has been previously calculated as in (11) by using the matrix inversion lemma.

Finally, the matrix inversion lemma can be applied again to calculate $\mathbf{R}^{-1}_{N-1}(n)$ from $(\overline{\mathbf{R}})^{-1}_N(n)$ and the previously calculated values of $\mathbf{r}_{N-1}(n)$ and $r_{N-1}(n)$. This method is described as follows:

1. $\boldsymbol{\alpha}_2(n) = (\overline{\mathbf{R}})^{-1}_N(n)\mathbf{r}_{N-1}(n)$
2. $\mathbf{R}^{-1}_{N-1}(n) = (\overline{\mathbf{R}})^{-1}_N(n) - \boldsymbol{\alpha}_2(n)[r_{N-1}(n) + \mathbf{r}^T_{N-1}(n)\boldsymbol{\alpha}_2(n)]^{-1}\boldsymbol{\alpha}^T_2(n)$.

The number of multiplications needed to calculate matrix $\mathbf{R}^{-1}_{N-1}(n)$ from $\mathbf{R}^{-1}_N(n-1)$ includes: the computation of $\mathbf{R}^{-1}_N(n)$, which has a cost of $4(N + N^2)$ multiplications, the calculation of $\mathbf{r}_{N-1}(n)$ from (14) and $r_{N-1}(n)$ from (15), which requires $2N$ multiplications, and finally the application of the matrix inversion lemma with a cost of $2N^2 - 2N$ multiplications. Therefore, the total number of multiplications required is $6N^2 + 4N$.

3.3. Iterative calculation of $\mathbf{R}_{N+1}^{-1}(n)$ *from* $\mathbf{R}_N^{-1}(n-1)$

We can consider in this case that

$$\mathbf{R}_{N+1}(n) = \begin{pmatrix} r_N(n) & \mathbf{r}_N^T(n) \\ & \\ \mathbf{r}_N(n) & \mathbf{R}_N(n-1) \end{pmatrix} \tag{17}$$

where $r_N(n)$ and $\mathbf{r}_N(n)$ can be recursively calculated as

$$r_N(n) = r_N(n-1) + x^2(n) - x^2(n-L) \tag{18}$$

and

$$\mathbf{r}_N(n) = \mathbf{r}_N(n-1) + \mathbf{x}_N(n-1)x(n) - \mathbf{x}_N(n-L-1)x(n-L). \tag{19}$$

Let us define $\boldsymbol{\alpha}_3(n) = \mathbf{R}_N^{-1}(n-1)\mathbf{r}_N(n)$ and, making use again of expressions (B.1) and (B.2), it follows that

$$a(n) = r_N(n) - \boldsymbol{\alpha}_3^T(n)\mathbf{r}_N(n). \tag{20}$$

Since $\mathbf{R}_N^{-1}(n-1)$ is a symmetric matrix, (17) can be rewritten as

$$\begin{aligned} \mathbf{R}_{N+1}^{-1}(n) &= \begin{pmatrix} 1/a(n) & -\boldsymbol{\alpha}_3^T(n)/a(n) \\ & \\ -\boldsymbol{\alpha}_3(n)/a(n) & \mathbf{R}_N^{-1}(n-1) + \boldsymbol{\alpha}_3(n)\boldsymbol{\alpha}_3^T(n)/a(n) \end{pmatrix} \\ &= \begin{pmatrix} 0 & \mathbf{0}^T \\ & \\ \mathbf{0} & \mathbf{R}_N^{-1}(n-1) \end{pmatrix} + \widehat{\boldsymbol{\alpha}}_3(n)\widehat{\boldsymbol{\alpha}}_3^T(n)/a(n), \end{aligned} \tag{21}$$

with $\widehat{\boldsymbol{\alpha}}_3(n) = [1, -\boldsymbol{\alpha}_3^T(n)]^T$ and $\mathbf{0}$ is a zero column vector of size N.

Thus, it can be calculated $\mathbf{R}_{N+1}^{-1}(n)$ from $\mathbf{R}_N^{-1}(n-1)$ as follows:

1. $\mathbf{r}_N(n) = \mathbf{r}_N(n-1) + \mathbf{x}_N(n-1)x(n) - \mathbf{x}_N(n-L-1)x(n-L)$
2. $\boldsymbol{\alpha}_3(n) = \mathbf{R}_N^{-1}(n-1)\mathbf{r}_N(n)$
3. $r_N(n) = r_N(n-1) + x^2(n) - x^2(n-L)$
4. $a(n) = r_N(n) - \boldsymbol{\alpha}_3^T(n)\mathbf{r}_N(n)$
5. $\widehat{\boldsymbol{\alpha}}_3(n) = [1, -\boldsymbol{\alpha_3}^T(n)]^T$
6. $\mathbf{R}_{N+1}^{-1}(n) = \begin{pmatrix} 0 & \mathbf{0}^T \\ & \\ \mathbf{0} & \mathbf{R}_N^{-1}(n-1) \end{pmatrix} + \widehat{\boldsymbol{\alpha}}(n)\widehat{\boldsymbol{\alpha}}^T(n)/a(n)$

Finally, the total number of multiplications required reaches $2N^2+6N+4$.

4. Simulation Results

As previously described, the exact inverse matrices required by variable order AP algorithms can be recursively calculated with a low computational cost. These recursive calculations give an exact inverse when the initial values of the inverses are accurate enough. For this reason, the algorithm must start with a setup period of $N \cdot L$ iterations before beginning the recursive calculations. Under these conditions the behavior of the FExVAP algorithm is identical to the VAP algorithm apart from, obviously, its computational cost.

In order to test the performance of the proposed FExVAP algorithm compared to the VAP algorithm and the AP algorithm with $N = 10$, several simulations have been carried out. In addition, simulation results of the E-AP and its computationally efficient approach, the FExE-AP, have been also carried out and compared with the algorithms previously mentioned. The learning curves of the algorithms are calculated by $10\log\left[\frac{e^2(n)}{d^2(n)}\right]$. These curves as well as the number of multiplications per iteration required have been calculated and shown in Figs. 2 and 3. The maximum value of the projection order was $N = 10$, and $\mu_{Nup} = 1/2$ and $\mu_{Ndown} = 1/4$ for the variable AP algorithms. Simulations were performed using the basic adaptive filtering scheme shown in Fig. 1 where $d(n)$ was chosen as $x(n)$ filtered through a finite impulse response filter ($\mathbf{P}$) of 20 randomly chosen coefficients and the input signal $x(n)$ was a zero mean Gaussian random signal function. The size of the adaptive filter was fixed to 19 coefficients thus a non zero residual error was always assured. Fig. 2 illustrates the algorithm behavior when the filter used to generate the desired signal, $d(n)$, remains invariable. Learning curves depicted in Fig. 2 have been obtained by averaging over $3,000$ independent trials of $10,000$ iterations (but only the first $1,000$ iterations are shown).

On the other hand, performance of the algorithms when the filter changes during the simulations is shown in Figs. 4 and 5. In this case, learning curves in Fig. 4 have been obtained from $3,000$ independent realizations of $30,000$ iterations.

Figures 2-5 show that the VAP and its fast version exhibit the same learning curves and both outperform the AP algorithm ($N = 10$) in terms of multiplications required and final residual error. Furthermore, the FExVAP algorithm requires less multiplications than the original VAP, mainly for high projection orders. Regarding the E-AP, its efficient version (FExE-

AP) provides the same learning curve and it falls close to the VAP curve with a slightly poorer performance in terms of convergence speed and final residual error. It has to be noted that the AP algorithm used the maximum allowed projection order in these simulations, $N = 10$, therefore the speed of convergence of the AP algorithm was the maximum available. The VAP algorithm behaves as fast as the AP algorithm during transient periods since it is able to dynamically adjust its projection order. The comparative of the total number of multiplications of the five algorithms is shown in Table 1. These values comprises the multiplications needed to carry out the total number of iterations of each algorithm ($10,000$ iterations for stationary and $30,000$ for non stationary environments). It can be seen that the FExVAP algorithm needs less multiplications than the VAP as well as the FExE-AP needs less multiplications than the E-AP. The computational cost reduction is more significant when the algorithm consumes more time using higher orders.

5. Conclusions

An exact and computationally efficient method to calculate the inverse signal matrices involved in the AP and variable order AP algorithms have been described and validated by simulations. Thus a fast exact variable order AP (FExVAP) algorithm has been developed. This algorithm outperforms the AP algorithm in terms of computational complexity, convergence speed and final residual error, and outperforms the VAP in number of multiplications mainly when the algorithm is working at high projection orders, which is frequent in non stationary environments. The developed recursive calculation of the inverse matrices can be used when $N \to N$, $N \to N+1$ or $N \to N-1$ between successive algorithm iterations.

References

[1] K. Ozeki and T. Umeda, "An adaptive filtering algorithm using an orthogonal projection to an affine subspace and its properties," *Proc. of the Electron. and Communic. in Japan*, vol. J67-A, pp. 126–132, Feb. 1984.

[2] A. H. Sayed, *Fundamentals of Adaptive Filtering, Wiley, New York*, 2003.

[3] B. Widrow and S. D. Stearns, *Adaptive Signal Processing,* Englewood Cliffs, N.J: Prentice-Hall, 1985.

[4] A. Carini, and G. Sicuranza, "Transient and steady-state analysis of filtered-x affine projection algorithm," *IEEE Trans. on Signal Process.*, vol. 54, no. 2, pp. 665–678, Feb. 2006.

[5] M. Bouchard, "Multichannel affine and fast affine projection algorithms for active noise control and acoustic equalization systems," *IEEE Trans. on Speech and Audio Proces.*, vol. 11, no. 1, pp. 54–60, Jan. 2003.

[6] J. Benesty, P. Duhamel, and Y. Granier, "A multichannel affine projection algorithm with applications to multichannel acoustic echo cancellation," *IEEE Signal Process. Lett.*, vol. 3, no. 2, pp. 35–37, Feb. 1996.

[7] G. Carayannis, D. G. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least-squares filtering and prediction," *IEEE Trans. on Acoustics, Speech and Signal Process.*, vol. 31, pp. 1394–1402, 1983.

[8] H. C.-Shin, A. H. Sayed and W.-J. Song "Variable step-size NLMS and affine projection algorithms," *IEEE Signal Process. Lett.*, vol. 11, no. 2, pp. 132–135, Feb. 2004.

[9] L. Liu, M. Fukumoto, S. Saiki, and S. Zhang, "A variable step-size proportionate affine projection algorithm for identification of sparse impulse response," *Eurasip J. on Advances in Signal Process.*, pp. 1–10, 2009.

[10] L. R. Vega, H. Rey, and J. Benesty, "A robust variable step-size affine projection algorithm," *Signal Processing*, vol. 90, no. 9, pp. 2806–2810, Sep. 2010.

[11] K. Mayyas, "A variable step-size affine projection algorithm," *Digital Signal Processing*, vol. 20, no. 2, pp. 502–510, Mar. 2010.

[12] S. J. Kong, K. Y. Hwang and W. J. Song, "An affine projection algorithm with dynamic selection of input vectors," *IEEE Signal Process. Lett.*, vol. 14, no. 8, pp. 529–532, Aug. 2007.

[13] S.-E. Kim, S.-J. Kong and W.-J. Song, "An affine projection algorithm with evolving order," *IEEE Signal Process. Lett.*, vol. 16, no. 11, pp. 937–940, Nov. 2009.

[14] N. W. Kong J. W. Shin and P. G. Park, "A two-stage affine projection

algorithm with mean square error matching step sizes," *Signal Processing*, vol. 91, no. 11, pp. 2639–2646, Nov. 2011.

[15] A. Gonzalez, M. Ferrer, M. de Diego, G. Piñero, and J. J. Lopez, "Practical implementation of multichannel adaptive filters based on FTF and AP algorithms for active control," *Int. J. Adapt. Control Signal Process.*, vol. 19, pp. 89–105, 2005.

[16] S. L. Gay and S. Tavathia, "The fast affine projection algorithm," *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, vol. 5, pp. 3023–3026, May 1995, Detroit, MI.

[17] M. Tanaka, Y. Kaneda, S. Makino and J. Kojima, "Fast projection algorithm and its step size control," *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.(ICASSP)*, vol. 2, pp. 945–948 May 1995.

[18] H. Ding, "A stable fast affine projection adaptation algorithm suitable for low-cost processors," *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.(ICASSP)*, vol. 1, pp. 360–363 Aug. 2000.

[19] Y. Zakharov, "Low complexity implementation of the affine projection algorithm," *IEEE Signal Process. Lett.*, vol. 15, pp. 557-560, 2008.

[20] F. Albu and H.K. Kwan, "Fast block exact Gauss-Seidel pseudo affine projection algorithm", *IEE Elect. Lett.*, Vol. 40, Issue:22, pp. 1451-1453, Oct. 2004.

[21] M. C. Tsakiris and P. A. Naylor, "Fast exact affine projection algorithm using displacement structure theory", in *Proc. of DSP 2009*, pp. 69-74, Jul. 2009.

[22] S. Haykin, *Adaptive Filter Theory,* Prentice-Hall, Ed., Upper Saddle River, NJ, fourth edition, 2002.

[23] F. Albu, C. Paleologu and J. Benesty, "A Variable Step Size Evolutionary Affine Projection Algorithm," *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Process.(ICASSP)*, pp. 429-432, May 2011, Prague, Czech Republic.

[24] C. Paleologu, J. Benesty, and S. Ciochina, "A variable step-size affine projection algorithm designed for acoustic echo cancellation," *IEEE Trans. Audio, Speech and Language Process.*, vol. 16, no. 8, pp. 1466–1478, Nov. 2008.

[25] S. G. Sankaran and A. A. Louis Beex, "Convergence behavior of affine projection algorithms," *IEEE Tran. on Signal Process.*, vol. 45, no. 4, pp. 1086–1096, Apr. 2000.

[26] M. Ferrer, M. de Diego, A. Gonzalez, and G. Piñero, "Efficient implementation of the affine projection algorithms for active noise control aplication," *Proc. 12th European Signal Process. Conf. (EUSIPCO)*, Sep. 2004.

[27] Y. Kaneda, M. Tanaka and J. Kojima, "An adaptive algorithm with fast convergence for multi-input sound control", *Proc. Active 95*, pp. 993-1004, Jul. 6-8, 1995.

Algorithm 1 FExVAP algorithm.

```
Input: Reference signal $x(n)$, matrix $\mathbf{R}_N^{-1}(n-1)$, and vectors $\mathbf{r}_N(n-1)$ and
    $\mathbf{r}_{N-1}(n-1)$
Output: $\mathbf{R}_N^{-1}(n)$ at time $n$
 1: if $N(n-1) = N(n)$ then
 2:     Update the vectors $\mathbf{x}_N(n)$ and $\mathbf{x}_N(n-L)$
 3:     $\boldsymbol{\alpha}_1(n) = \mathbf{R}_N^{-1}(n-1)\mathbf{x}_N(n)$
 4:     $\mathbf{Q}_N^{-1}(n) = \mathbf{R}_N^{-1}(n-1) - \boldsymbol{\alpha}_1(n)\boldsymbol{\alpha}_1^T(n)/[1+\mathbf{x}_N^T(n)\boldsymbol{\alpha}_1(n)]$
 5:     $\boldsymbol{\beta}(n) = \mathbf{Q}_N^{-1}(n)\mathbf{x}_N(n-L)$
 6:     $\mathbf{R}_N^{-1}(n) = \mathbf{Q}_N^{-1}(n) + \boldsymbol{\beta}(n)\boldsymbol{\beta}^T(n)/[1-\mathbf{x}_N^T(n-L)\boldsymbol{\beta}(n)]$
 7: else if $N(n-1) = N(n)+1$ then
 8:     Update the vectors $\mathbf{x}_{N-1}(n)$ and $\mathbf{x}_{N-1}(n-L)$
 9:     Compute $\mathbf{R}_N^{-1}(n)$ as in the case $N(n-1) = N(n)$
10:     Derive $(\overline{\mathbf{R}})_N^{-1}(n)$ from $\mathbf{R}_N^{-1}(n)$
11:     $\mathbf{r}_{N-1}(n) = \mathbf{r}_{N-1}(n-1) + \mathbf{x}_{N-1}(n)x(n-N+1) - \mathbf{x}_{N-1}(n-L)x(n-N-L+1)$
12:     $r_{N-1}(n) = r_{N-1}(n-1) + x^2(n-N+1) - x^2(n-N-L+1)$
13:     $\boldsymbol{\alpha}_2(n) = (\overline{\mathbf{R}})_N^{-1}(n)\mathbf{r}_{N-1}(n)$
14:     $\mathbf{R}_{N-1}^{-1}(n) = (\overline{\mathbf{R}})_N^{-1}(n) - \boldsymbol{\alpha}_2(n)[r_{N-1}(n) + \boldsymbol{r}_{N-1}^T(n)\boldsymbol{\alpha}_2(n)]^{-1}\boldsymbol{\alpha}_2^T(n)$
15: else if $N(n-1) = N(n)-1$ then
16:     Update the vectors $\mathbf{x}_N(n)$ and $\mathbf{x}_N(n-L-1)$
17:     $\mathbf{r}_N(n) = \mathbf{r}_N(n-1) + \mathbf{x}_N(n-1)x(n) - \mathbf{x}_N(n-L-1)x(n-L)$
18:     $\boldsymbol{\alpha}_3(n) = \mathbf{R}_N^{-1}(n-1)\mathbf{r}_N(n)$
19:     $r_N(n) = r_N(n-1) + x^2(n) - x^2(n-L)$
20:     $a(n) = r_N(n) - \boldsymbol{\alpha}_3^T(n)\mathbf{r}_N(n)$
21:     $\widehat{\boldsymbol{\alpha}}_3(n) = [1, -\boldsymbol{\alpha}_3^T(n)]^T$
22:     $\mathbf{R}_{N+1}^{-1}(n) = \begin{pmatrix} 0 & \mathbf{0}^T \\ & \\ \mathbf{0} & \mathbf{R}_N^{-1}(n-1) \end{pmatrix} + \widehat{\boldsymbol{\alpha}}_3(n)\widehat{\boldsymbol{\alpha}}_3^T(n)/a(n)$
23: end if
```

Algorithm	Mult-1	Mult-2
AP (N=10)	$29.47X_1$	$5.12X_2$
VAP	$1.30X_1$	$1.88X_2$
FExVAP	X_1	X_2
E-AP	$1.64X_1$	$1.93X_2$
FExE-AP	$1.34X_1$	$1.05X_2$

Table 1: Comparative total multiplications for stationary (Mult-1) and non stationary (Mult-2) environments.

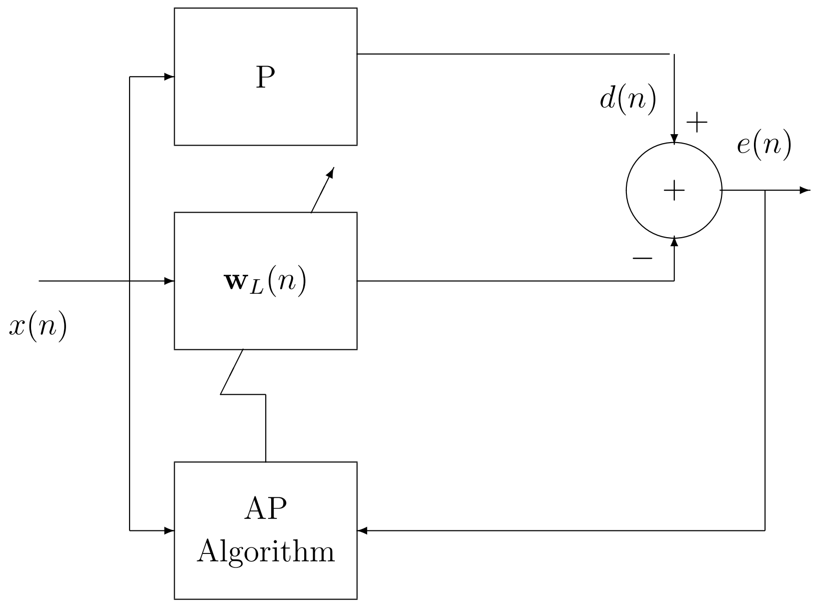


Figure 1: Basic adaptive system identification scheme.

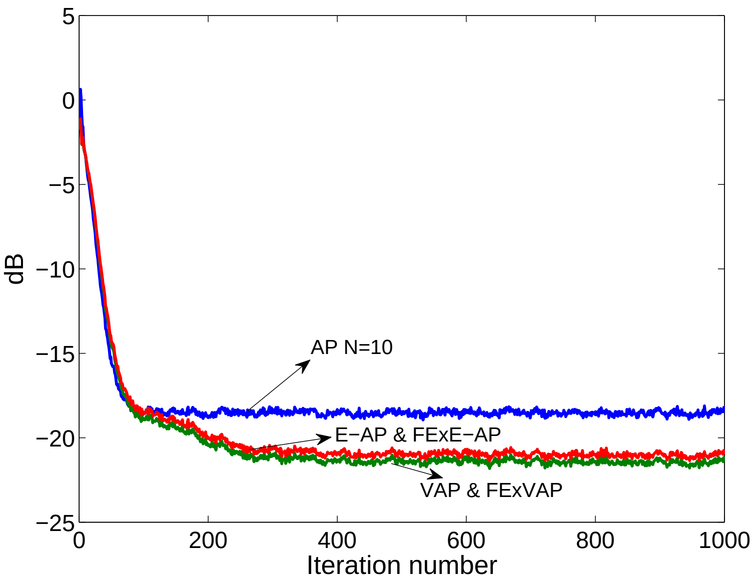
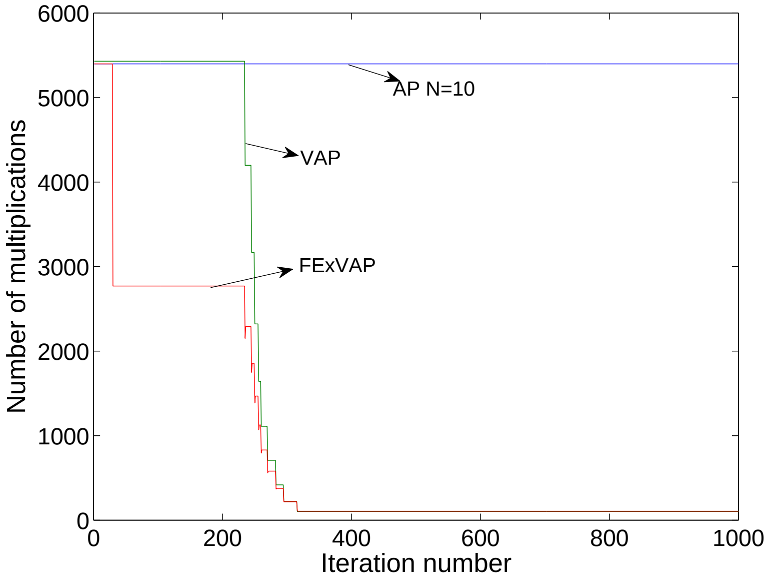
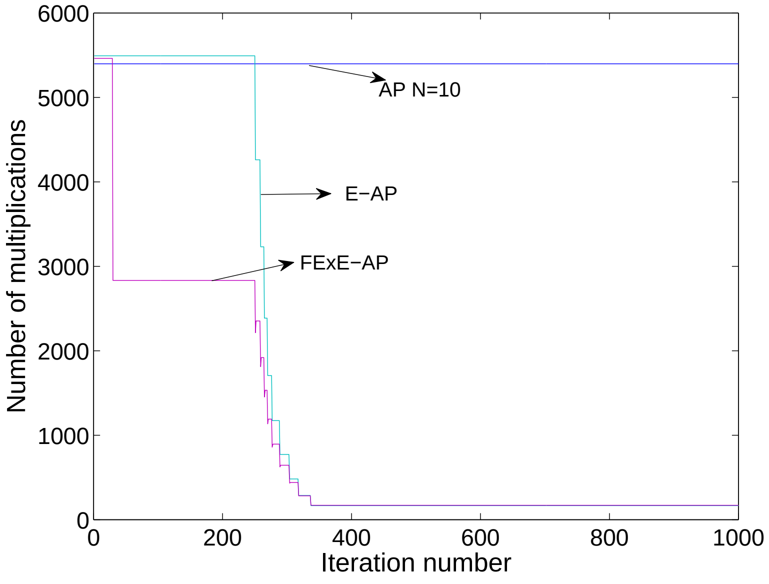


Figure 2: Learning curves for the AP (N=10), the VAP, the E-AP, and their fast exact approaches (FExVAP and FExE-AP) for a stationary environment during the first 1000 iterations.



(a)



(b)

Figure 3: Number of multiplications per iteration for the AP (N=10), (a) the VAP and the FExVAP, and (b) the E-AP and the FExE-AP in a stationary environment during the first 1000 iterations.

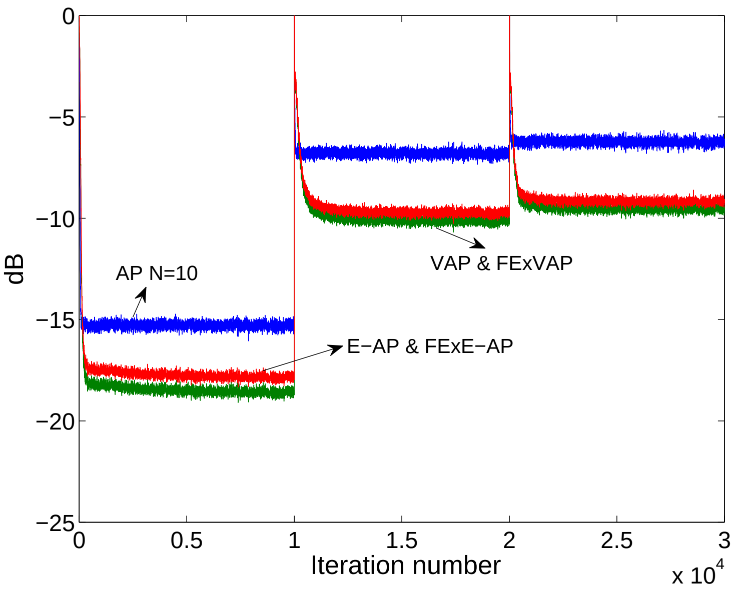
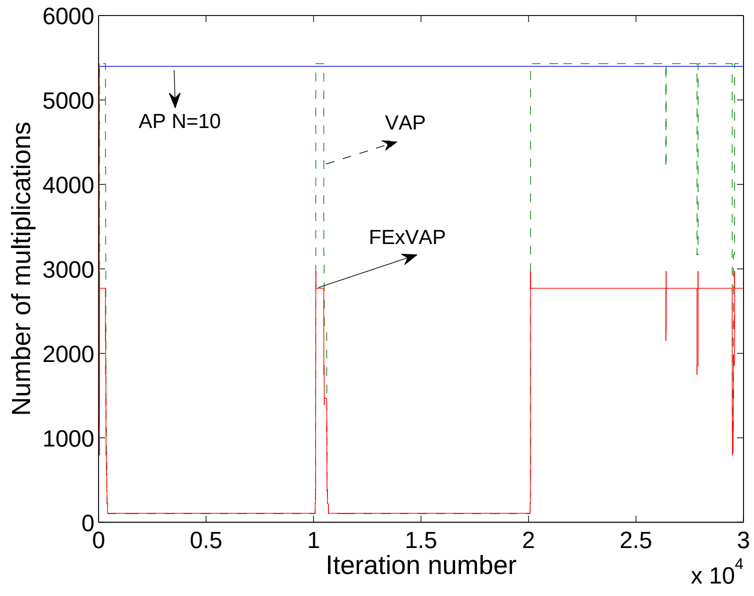
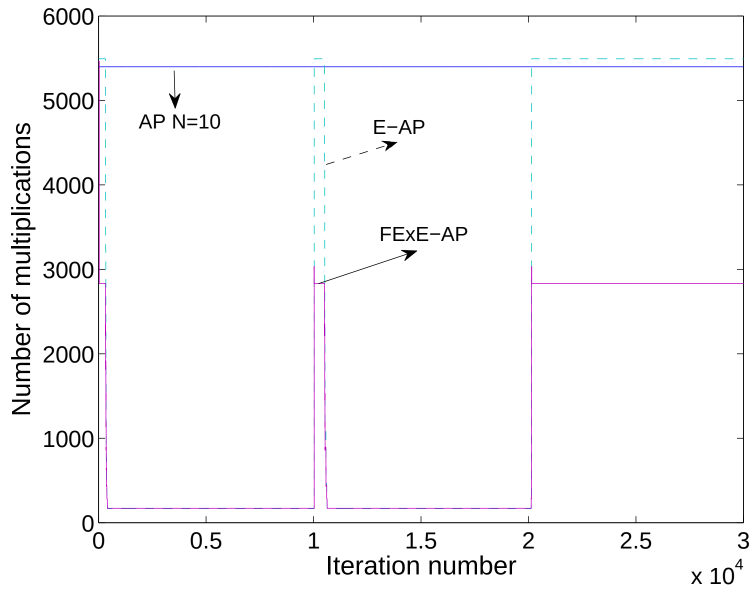


Figure 4: Learning curves for the AP (N=10), the VAP, the E-AP, and their fast exact approaches (FExVAP and FExE-AP) in a non stationary environment.



(a)



(b)

Figure 5: Number of multiplications per iteration for the AP (N=10), (a) the VAP and the FExVAP, and (b) the E-AP and the FExE-AP in a non stationary environment.

Appendix A. Matrix inversion lemma

Let $\mathbf{\Gamma}$ and $\mathbf{\Theta}$ be two positive definite $N \times N$ matrices that fulfill, [22][27]

$$\mathbf{\Gamma} = \mathbf{\Theta}^{-1} + \mathbf{\Phi}\mathbf{\Psi}^{-1}\mathbf{\Phi}^T, \tag{A.1}$$

where $\mathbf{\Psi}$ is a $M \times M$ positive definite matrix and $\mathbf{\Phi}$ a $N \times M$ matrix, then, the inverse of $\mathbf{\Gamma}$ can be calculated as

$$\mathbf{\Gamma}^{-1} = \mathbf{\Theta} - \mathbf{\Theta}\mathbf{\Phi}(\mathbf{\Psi} + \mathbf{\Phi}^T\mathbf{\Theta}\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{\Theta}. \tag{A.2}$$

Appendix B. Matrix inversion in block form

It is known that an inverse matrix can be calculated from its parts by

$$\begin{pmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{pmatrix}^{-1} = \begin{pmatrix} \mathbf{F}_{11}^{-1} & -\mathbf{F}_{11}^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ -\mathbf{A}_{22}^{-1}\mathbf{A}_{21}\mathbf{F}_{11}^{-1} & \mathbf{A}_{22}^{-1} + \mathbf{A}_{22}^{-1}\mathbf{A}_{21}\mathbf{F}_{11}^{-1}\mathbf{A}_{12}\mathbf{A}_{22}^{-1} \end{pmatrix} \tag{B.1}$$

with

$$\mathbf{F}_{11} = \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}. \tag{B.2}$$