Fast Recursive AMIPAP Algorithm

Felix Albu
Faculty of Electronics
Valahia University of Targoviste
Targoviste, Romania
felix.albu@valahia.ro

***Abstract* – The approximate memory improved proportionate affine projection algorithm has been proposed for sparse system identification. This paper presents a fast recursive implementation of this algorithm. Three ideas used previously for other affine projection variants are used: auxiliary coefficients vectors, periodically update of the proportionate coefficients and recursive filtering of the error vector. Simulation results are made in order to show the performance of the algorithm for network echo cancellation example.**

Keywords-adaptive filtering, proportionate affine projection, sparse impulse response, fast implementation.

I. INTRODUCTION

The normalized least mean squares (NLMS) algorithm and the affine projection (AP) algorithms have been extensively used for network and acoustic echo cancellation. Their proportionate versions proved very useful especially for cases of long and sparse echo paths. The proportionate-type algorithms have an increased convergence rate because each coefficient of the filter is updated independently of the others, by adjusting the adaptation step-size in proportion to the magnitude of the estimated filter coefficient. The proportionate AP (PAP) [1] and improved PAP (IPAP) [2], the memory IPAP (MIPAP) algorithm [3], the μ-law MIPAP (MMIPAP) [4] and individual-activation-factor memory PAP (IAF-MPAP) [5] algorithms were developed starting from this principle. The main difference between the above mentioned algorithms is given by the proportionate matrix computation. The complexity of the memory PAP (MPAP) algorithms [6] can be reduced by using sign based proportionate affine algorithms in case of impulsive environments [7]. An approximated MIPAP (AMIPAP) algorithm [6] was proposed to reduce the complexity of MIPAP by forcing the symmetry of a matrix to be inverted. Several simplified proportionate and intermittently updated versions have been investigated in [8]-[9]. Also simplified or dichotomous coordinate descent [10] versions were developed [11].

In this paper, a fast version of AMIPAP-family type algorithms is developed by combining the proportionate matrix approximation from [6], the fast exact filtering approach from [6] and [11], the auxiliary coefficient vector computation taken from [12] and the periodic update of the proportionate coefficients experimented in [8] and [13]. It is also shown in this paper that AMIPAP algorithms and its fast recursive version need a higher regularization factor at beginning in order to compensate for the approximation of the proportionate matrix.

The paper is organized as follows. Section II presents the proposed algorithm and investigates its numerical complexity. The simulation results are presented in Section III. Finally, the conclusions are given and ideas for further improvements are proposed.

II. FRAMIPAP ALGORITHM

In an echo cancellation system, the far-end signal is $x(k)$, and the reference signal $d(k)$, where k is the time index. The adaptive FIR filter is given by the coefficient vector $\mathbf{w}(k)=\left[w_0(k)...w_{L-1}(k)\right]^T$, where L is the length of the adaptive filter and superscript T denotes transposition. The error signal vector is given by $\mathbf{e}(k)=\mathbf{d}(k)-\mathbf{y}(k)$, where the desired signal is $\mathbf{d}(k)=\left[d(k)...d(k-M+1)\right]^T$, the filtered-out vector is $\mathbf{y}(k)=\mathbf{X}^T(k)\mathbf{w}(k-1)=\left[y_0(k)...y_{M-1}(k)\right]^T$, the input data matrix is $\mathbf{X}(k)=\left[\mathbf{x}(k)...\mathbf{x}(k-M+1)\right]$, the input vector is $\mathbf{x}(k)=\left[x(k)...x(k-L+1)\right]$, and $\mathbf{e}(k)=\left[e_0(k)...e_{M-1}(k)\right]^T$, M is the projection order and filter length and L is the length of the adaptive filter.

The weight coefficients of AMIPAP algorithm are computed as follows [6]:

$$\mathbf{w}(k)=\mathbf{w}(k-1)+\mu\mathbf{P}(k)\mathbf{S}^{-1}(k)\mathbf{e}(k) \tag{1}$$

where μ is the normalized step-size parameter and $\mathbf{P}(k)$ and $\mathbf{S}(k)$ are computed as shown in the following lines. Like in [3]

$$\mathbf{P}(k)=\left[\mathbf{g}(k-1)\odot\mathbf{x}(k)\quad\mathbf{P}_{-1}(k-1)\right] \tag{2}$$

with $$\mathbf{P}_{-1}(k-1)=\begin{bmatrix}\mathbf{g}(k-2)\odot\mathbf{x}(k-1) & ... \\ \mathbf{g}(k-M)\odot\mathbf{x}(k-M+1)\end{bmatrix},$$

contains the first $M-1$ columns of $\mathbf{P}(k-1)$ and the operator $\odot$ denotes the Hadamard product.

The matrix $\mathbf{S}(k)$, is obtained by updating both its first row and its first column with $\mathbf{X}^T(k)\cdot\left[\mathbf{g}(k-1)\odot\mathbf{x}(k)\right]$ and adding δ to the first

element, where δ is a regularization parameter. The bottom-right $(M-1)\times(M-1)$ submatrix of $\mathbf{S}(k)$ is replaced with the top-left $(M-1)\times(M-1)$ submatrix of $\mathbf{S}(k-1)$ [6].

In AMIPAP algorithm the proportionate coefficients are computed as follows:

$$g_l(k)=\frac{1-\alpha}{2L}+(1+\alpha)\frac{|w_l(k)|}{2\sum_{i=0}^{L-1}|w_i(k-1)|+\theta} \quad (3)$$

where θ is a small constant and $\alpha\in[-1\ 1]$.

In case of μ-law algorithms [4] $g_l(k)$ is evaluated as follows:

$$g_l(k)=\frac{1-\alpha}{2L}+(1+\alpha)\frac{F(|w_l(k)|)}{2\sum_{i=0}^{L-1}F(|w_i(k-1)|)+\theta} \quad (4)$$

where $F(|w_l(k)|)=\ln(1+\mu_l|w_l(k)|)$ and μ_l is a constant.

The fast adaptation of the weight vector is inspired from [12] and adapted from [14]:

$$\hat{\mathbf{w}}(k+1)=\hat{\mathbf{w}}(k)+(\mathbf{g}(k-M)\odot\mathbf{x}(k-M+1))E_{M-1}(k) \quad (5)$$

and

$$\mathbf{E}(k)=\boldsymbol{\varepsilon}(k)+\begin{bmatrix}0\\ \bar{\mathbf{E}}(k-1)\end{bmatrix} \quad (6)$$

where $\bar{\mathbf{E}}(k-1)$ consists of upper $M-1$ elements of $\mathbf{E}(k)=[E_0(k),...,E_{M-1}(k)]^T$ whose elements are given by $E_m(k)=\sum_{i=0}^{m}\varepsilon_i(k-m+i)$ [13] and

$$\begin{aligned}\boldsymbol{\varepsilon}(k)&=[\varepsilon_o(k),...,\varepsilon_{M-1}(k)]^T\\ &=\mu\mathbf{P}(k)\mathbf{S}^{-1}(k)\mathbf{e}(k)\end{aligned} \quad (7)$$

Another step in deriving the fast implementation of AMIPAP is the use of a fast filtering procedure [10]. This step was firstly used for proportionate algorithms in [11]. Following the same steps as in [11] the following equations are obtained:

$$\mathbf{z}(k)=\begin{bmatrix}\mathbf{x}^T(k)\hat{\mathbf{w}}(k-1)+\mathbf{r}^T(k)\mathbf{E}(k-2)\ ...\\ y^0(k-1)\ ...\ y^{M-2}(k-1)\end{bmatrix}^T \quad (8)$$

$$\mathbf{y}(k)=\mathbf{z}(k)+\mathbf{G}(k)\varepsilon(k-1) \quad (9)$$

where $\mathbf{G}(k)=\mathbf{X}^T(k)\mathbf{P}(k-1)$ and $\mathbf{r}(k)=\mathbf{P}^T(k-2)\mathbf{x}(k)$ [13].

The update of $\mathbf{G}(k)$ and $\mathbf{r}(k)$ requires few operations due to their special structure and common elements with other matrices and vectors computed in previous iterations.

It can be noticed that the auxiliary coefficients $\hat{\mathbf{w}}(k)$ are updated instead of the real coefficients $\mathbf{w}(k)$ that are needed to compute the proportionate coefficients vector $\mathbf{g}(k)$. In order to reduce the overall average complexity these coefficients can be computed less often. It was found by simulations that this approximation does not lead to important performance losses if the update factor is around the projection factor *M* for typical values (i.e., between 2 and 10). This finding is in accordance with previous findings regarding the intermittently updated proportionate affine projection algorithms [8], [9]. In this case, the computation of **g**(*k*) needs *LM/p* multiplications per sample, where *p* is the updating factor. If an update is performed at every *M* iterations (i.e., *p* = *M*) the computation of **g**(*k*) needs *L* multiplications per sample. The proposed algorithm is called Fast Recursive AMIPAP (FRAMIPAP) algorithm. In case the logarithmic coefficients are used the acronym is FRAMMIPAP. The AMIPAP using logarithmic coefficients is called AMMIPAP.

A. *Numerical complexity*

The numerical complexity of FRAMIPAP in terms of multiplications is the following (for *p* = *M*):

$$C_{\text{FRAMIPAP}}=L(M+6+1/M)+M^2+M+P_m \quad (10)$$

The notation $P_m = O(M^3)$ indicates the numerical complexity in terms of multiplications associated with solving the linear systems of equations using the **LDL**^{T} method [15]. The MIPAP complexity in terms of multiplications is $C_{\text{MIPAP}}=L(4M+1)+M+P_m$ and the complexity of AMIPAP is $C_{\text{AMIPAP}}=L(3M+2)+M+P_m$ [6]. The use of logarithmic proportionate coefficients requires additional *L* logarithmic functions and *L* additions per iteration in comparison with the linear coefficients.

Fig. 1 shows the numerical complexity comparison in two situations: a) as a function of *L* and fixed *M* = 10 and b) as a function of *M* and fixed $L=512$. The number of multiplications varies linearly with the filter length for all the considered algorithms. It can be seen from Fig. 1 that FRAMIPAP is the least complex in terms of multiplications, followed by AMIPAP and MIPAP. The computational savings are higher especially for large filter lengths or projection orders.

For example, with $L = 512$ and $M = 10$, the FRAMIPAP needs only 8609 multiplications, while the MIPAP needs 21257 multiplications. Therefore, MIPAP is about 2.5 times more complex than the proposed algorithm. AMIPAP needs 16394 multiplications, therefore, it is about 90% more complex than FRAMIPAP. This important complexity reduction in terms of multiplications is compensated by increased memory requirements.

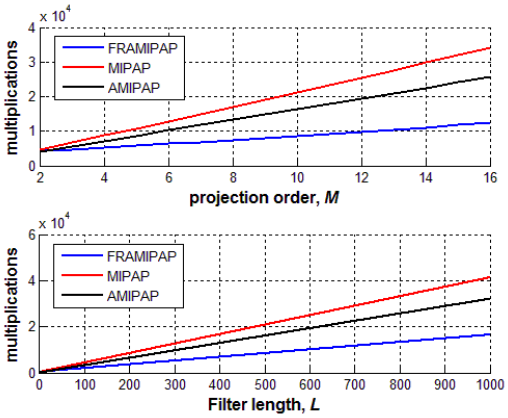

Fig. 1 Numerical complexity of the considered algorithms in terms of multiplications for two situations: a) $L = 512$, variable M; b) $M = 10$, variable L.

III. Simulation Results

Computer simulations are made for a network echo cancellation application. The sampling rate is 8 kHz and the first impulse response from ITU-T G.168 padded with zero is used [16]. The other parameters are $L = 512$, and SNR = 30 dB. An abrupt change of the echo path is simulated by shifting the impulse response with 12 samples. The performance is evaluated in terms of mean square deviation (MSD) and the results are averaged over 10 trials. The step size was set to $\mu = 0.5$, $\alpha = 0$, and $\mu_l = 1000$ were used.

For the first simulation an AR(1) with coefficients (1, -0.95) is used. The MSD results of FRAMIPAP and AMIPAP (corresponding to FRAMIPAP without periodic update) were compared. The regularization parameter was $\delta = 200\sigma_x^2 / L$ and $M = 10$. It can be seen from Fig. 2a that the performance of both algorithms are close and the MSD difference is less than 1dB for this case.

The AMIPAP algorithm has a high sensitivity in case of lower regularization values [13]. This is caused by the approximation used in deriving the algorithm. Even when using very small regularization values a simple solution is to use a much higher regularization value for the first 100-300 iterations and return to the initial value afterwards. The number of iterations depends on the filter length. Fig. 3a shows the MSD difference between MIPAP and AMIPAP using a very small regularization value (as simulated in [13]) and a stabilized AMIPAP. The input was a composite source signal (CSS) and $M = 4$. Fig. 3b clearly shows that the eventual spikes of the AMIPAP convergence curve that appears when using a very low regularization value has disappeared when a much higher initial regularization value was used. The stabilized solution was used in all the other simulations of this paper for the AMIPAP or FRAMIPAP algorithms.

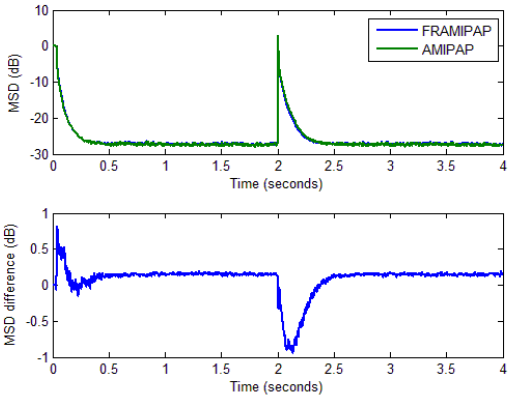

Fig. 2 a) MSD performance of the FRAMIPAP and AMIPAP algorithms for an AR(1) input; b) MSD difference.

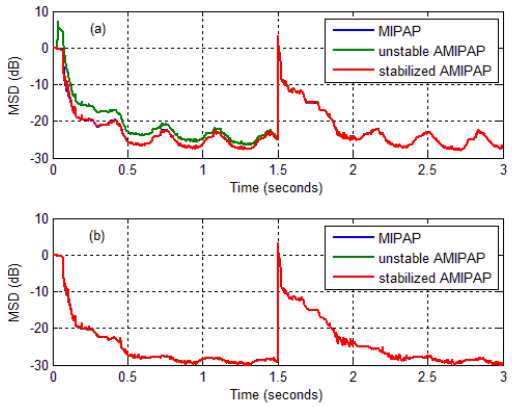

Fig. 3 MSD performance of MIPAP, unstable AMIPAP and stabilized AMIPAP for a CSS input and two situations: a) $\delta = 10\sigma_x^2 / L$; b) $\delta = 300\sigma_x^2 / L$.

In Fig. 4, the performance of the proposed FRAMIPAP is compared with that of MIPAP and AMIPAP algorithms for both linear and logarithmic proportionate coefficients [4] and $M = 5$.

It can be observed from Fig. 4 that the proposed fast recursive algorithms achieve similar performance with their original versions. A comparison of the original algorithms performance with other state-of-the-art algorithms has been investigated in [3], [6], [8], [14] etc. Therefore, it is proved that an update at every M iterations of the proportionate vector is not degrading the performance of the proposed algorithms for both computational methods (linear and logarithmic respectively). The possible few dB losses is a good compromise for the important complexity reduction achieved by the proposed fast recursive versions. However, special attention should be given to the regularization values and high regularization values are required for a good convergence behavior.

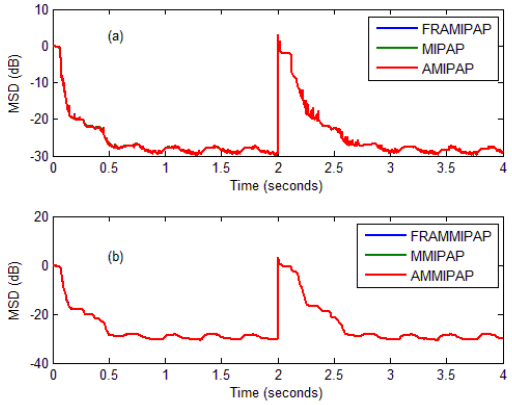

Fig. 4 Misalignment performance of the considered algorithms with $M = 5$ and $\mu = 0.5$ for speech input; a) linear proportionate coefficients; b) logarithmic proportionate coefficients.

Our simulations have shown that if a very high projection order is used (i.e., $M > 10$) a lower update factor should be used for very sparse echo paths (i.e., $p < 10$).

The performance of the proposed algorithm using 32-bit logarithmic number system (LNS) implementation [17] was investigated and compared with the 32-bit floating point results. The results are almost identical and confirmed similar conclusions obtained for a wide range of algorithms [17].

Future work will be focused on developing an intermittently update version depending on the detected path sparseness or filter state as in [8] or [9] and adapt it for active noise control [18] and adaptive feedback cancellation [19].

IV. CONCLUSION

In this paper a low-complexity implementation of AMIPAP using both linear and logarithmic proportionate coefficients has been presented. The time-shift and symmetry of several matrices and an equivalent periodic update of the filter weights has been used in order to derive the FRAMIPAP algorithm. It is shown that an important complexity reduction is achieved with minor performance losses if special care is given to the choice of the regularization factor.

ACKNOWLEDGMENT

This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-II-ID-PCE-2011-3-0097.

REFERENCES

[1] D. L. Duttweiler, "Proportionate normalized least mean-squares adaptation in echo cancellers," IEEE Trans. Speech Audio Process., Feb. 2000, vol. 8, pp. 508–518.

[2] O. Hoshuyama, R. A. Goubran, and A. Sugiyama, "A generalized proportionate variable step-size algorithm for fast changing acoustic environments," in Proc. of IEEE ICASSP 2004,vol. IV, pp. 161–164.

[3] C. Paleologu, S. Ciochina, and J. Benesty, "An efficient proportionate affine projection algorithm for echo cancellation," IEEE Signal Process. Lett., Feb. 2010, vol. 17, pp. 165–168.

[4] J. Yang and G. E. Sobelman, "Efficient μ-law improved proportionate affine projection algorithm for echo cancellation," Electron. Lett., Jan. 2011, vol. 47, pp. 73–74.

[5] H. Zhao, Y. Yu, S. Gao, X. Zeng, and Z. He, "Memory proportionate APA with individual activation factors for acoustic echo cancellation," IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 22, pp. 291–294, June 2014.

[6] F. Albu, C. Paleologu, J. Benesty, and S. Ciochina, "A low complexity proportionate affine projection algorithm for echo cancellation," in Proc. of EUSIPCO 2010, pp. 6–10.

[7] F. Albu, H.K. Kwan, "Memory improved proportionate affine projection sign algorithm," IET Electronics Letters, vol. 48, (20), October 2012, pp. 1279-1281.

[8] F. Albu, "New proportionate affine projection algorithm," in Proc. of Inter-Noise and Noise-Con Congress and Conference, 2012, vol. 11, pp. 40-46.

[9] F. Albu, H. Coanda, D. Coltuc and M. Rotaru, "Intermittently updated simplified proportionate affine projection algorithm," in Proc. of ADAPTIVE 2014, pp. 42-47.

[10] Y. Zakharov, "Low complexity implementation of the affine projection algorithm," IEEE Signal Processing Letters, vol. 15, pp. 557-560, 2008.

[11] F. Albu, "A proportionate affine projection algorithm using fast recursive filtering and dichotomous coordinate descent iterations," in Proc. of SPAMEC 2011, August 2011, pp. 93-96

[12] S. Gay, S. Tavathia, "The fast affine projection algorithm," in Proc. of ICASSP'95, 1995, pp. 3023–3026.

[13] F. Yang, J. Yang, "Fast implementation of a family of memory proportionate affine projection algorithm," in Proc. of ICASSP 2015, pp. 3507-3511.

[14] F. Yang, M. Wu, J. Yang, and Z. Kuang, "A fast exact filtering approach to a family of affine projection-type algorithms," Signal Processing, August 2014, vol. 101, pp. 1–10.

[15] G. H. Golub and C. F. Van Loan, Matrix computation, 3rd edition. Baltimore, MD: The John Hopkins Univ. Press, 1996.

[16] Digital Network Echo Cancellers, ITU-T Rec. G.168, 2002.

[17] F. Albu, J. Kadlec, N. Coleman, A. Fagan, "Pipelined Implementations of the *A Priori* Error-Feedback LSL Algorithm Using Logarithmic Number System," in Proc. Of ICASSP 2002, May 2002, pp. 2681-2684.

[18] M. Bouchard, F. Albu, "The Gauss-Seidel fast affine projection algorithm for multichannel active noise control and sound reproduction systems," International Journal of Adaptive Control and Signal Processing, vol. 19, nr. 2-3, pp. 107-123, March-April 2005.

[19] M. Rotaru, F. Albu, H. Coanda, "A Variable Step Size Modified Decorrelated NLMS Algorithm for Adaptive Feedback Cancellation in Hearing Aids," in Proc. of ISETC 2012, Nov. 2012, pp. 263-266.