A NEW TWO-DIMENSIONAL BLOCK LEAST MEAN SQUARES ADAPTIVE ALGORITHM

S. Attallah and M. Najim
Equipe Signal/Image and GDR-134-CNRS
ENSERB. Av. du Dr. Albert Schweitzer
BP 99 - 33402 Talence Cedex
FRANCE
e-mail: attallah@goelette.tsi.u-bordeaux.fr

ABSTRACT

In this paper, a new 2-D block LMS algorithm is presented. This algorithm, which is an exact mathematical formulation of classical 2-D LMS algorithms, presents the advantage of preserving a good convergence as the block size increases. The reduction in the computational complexity is achieved by exploiting the redundancy between successive computations, rather than using disjoint or partially overlapping windows. The latter are known to degrade the convergence when the block size is large.

1 INTRODUCTION

During these last years, 2-D adaptive filters have received a great deal of attention. Their applications include, among others, image enhancement, deblurring and restoration. Most of 2-D adaptive algorithms have been extended from their 1-D counterpart, such as the 2-D LMS algorithm [1] which is the 2-D extension of the classical 1-D LMS algorithm [5]. The same is true for 2-D block LMS algorithms. In 1-D domain, J. Benesty and P. Duhamel [2] developed the 1-D exact block LMS whose computational complexity has been reduced by removing the redundancy in the calculations. They showed that it outperforms classical 1-D block LMS algorithms. In this paper, we develop the 2-D counterpart of this algorithm. We will call it the Fast Exact Two-Dimensional LMS (FETDLMS) algorithm, because it is mathematically equivalent to the 2-D LMS algorithm [1], while classical 2-D block LMS algorithms [4] are not. The method, we will be discussing, consists of transforming the 2-D FIR filtering operation in the algorithm into a set of 1-D FIR filtering operations which can be then carried out using fast 1-D filtering techniques.

2 FAST EXACT TWO-DIMENSIONAL LMS (FETDLMS) ALGORITHM

The two-dimensional LMS algorithm is given by the following equations [1]:

$$e(j) = y(m,n) - \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} H_j(l,k)\, X(m-l,n-k) \tag{1}$$

$$H_{j+1}(l,k) = H_j(l,k) + 2\mu e(j) X(m-l,n-k) \tag{2}$$

where X is the input image data window, H_j is the transversal filter window, y(m,n) is the (m,n)th element of the desired image and e(j) represents the error between the desired and the estimated images at the jth iteration. At iteration (j-1), equations (1) and (2) become

$$e(j-1) = y(m,n-1) - \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} H_{j-1}(l,k)\, X(m-l,n-1-k) \tag{1-a}$$

$$H_j(l,k) = H_{j-1}(l,k) + 2\mu e(j-1) X(m-l\ ,n-1k) \tag{2-a}$$

Substituting equation (2-a) into equation (1) leads to

$$\begin{aligned} e(j) = y(m,n) &- \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} H_{j-1}(l,k)\, X(m-l,n-k) \\ &- 2\mu e(j-1) \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} X(m-l,n-k)\, X(m-l,n-1-k) \end{aligned} \tag{3}$$

Now, let us set S(m,n), H_j and x_j as

$$S(m,n) = 2\mu \sum_{l=0}^{N-1} \sum_{k=0}^{N-1} X(m-l,n-k)\, X(m-l,n-1-k) \tag{4}$$

$$H_j = \begin{bmatrix} H_j(0,0) & \cdots & H_j(0,N-1) \\ \vdots & & \vdots \\ H_j(N-1,0) & \cdots & H(N-1,N-1) \end{bmatrix} \tag{5}$$

$$x_j = \begin{bmatrix} X(m,n) & \cdots & X(m,n-N+1) \\ \vdots & & \vdots \\ X(m-N+1,n) & \cdots & X(m-N+1,n-N+1) \end{bmatrix} \tag{6}$$

and let us define x_{H} and h_j as column vectors containing, respectively, the ith row of the data matrix xj and the ith row of the filter window, i.e.,

$$x_H(m-i,n) = \left[X(m-i,n)\ X(m-i,n-1)\ \ldots\ X(m-i,n-N+1)\right]^T \tag{7}$$

and

$$h_j(i) = \left[H_j(i,0)\ H_j(i,1)\ \ldots\ H_j(i,N-1)\right]^T \tag{8}$$

where i is a positive integer which takes the values from 0 to N-1 and superscript T denotes vector transposition. After gathering equations (1) and (1-a) into the same matrix and transforming the 2-D convolution in each one into a sum of 1-D convolutions, we can get equation (9) given just below.

$$\begin{bmatrix} e(j-1) \\ e(j) \end{bmatrix} = \begin{bmatrix} y(m,n-1) \\ y(m,n) \end{bmatrix} - \begin{bmatrix} x_H^T(m,n-1) \\ x_H^T(m,n) \end{bmatrix} h_{j-1}(0) - \begin{bmatrix} x_H^T(m-1,n-1) \\ x_H^T(m-1,n) \end{bmatrix} h_{j-1}(1) - \ldots - \begin{bmatrix} x_H^T(m-N+1,n-1) \\ x_H^T(m-N+1,n) \end{bmatrix} h_{j-1}(N-1) - \begin{bmatrix} 0 & 0 \\ S(m,n) & 0 \end{bmatrix} \begin{bmatrix} e(j-1) \\ e(j) \end{bmatrix} \quad (9)$$

$$\begin{bmatrix} e(j-1) \\ e(j) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -S(m,n) & 1 \end{bmatrix} \left(\begin{bmatrix} y(m,n-1) \\ y(m,n) \end{bmatrix} - \begin{bmatrix} x_H^T(m,n-1) \\ x_H^T(m,n) \end{bmatrix} h_{j-1}(0) - \begin{bmatrix} x_H^T(m-1,n-1) \\ x_H^T(m-1,n) \end{bmatrix} h_{j-1}(1) - \ldots - \begin{bmatrix} x_H^T(m-N+1,n-1) \\ x_H^T(m-N+1,n) \end{bmatrix} h_{j-1}(N-1) \right) \quad (10)$$

This last equation can be further arranged to give equation (10). The reduction in the computational complexity of each 1-D convolution can be achieved by applying fast filtering techniques [2] as follows:

$$\begin{bmatrix} x_H^T(m-i,n-1) \\ x_H^T(m-i,n) \end{bmatrix} h_{j-1}(i) = \begin{bmatrix} A_{(i,1)} & A_{(i,2)} \\ A_{(i,0)} & A_{(i,1)} \end{bmatrix} \begin{bmatrix} W_{(i,0)}(j-1) \\ W_{(i,1)}(j-1) \end{bmatrix} \quad (11)$$

Where

$$A_{(i,0)} = [X(m-i,n)\ X(m-i,n-2)\ X(m-i,n-4)\ \ldots\ X(m-i,n-N+2)]^T \quad (12)$$

$$A_{(i,1)} = [X(m-i,n-1)\ X(m-i,n-3)\ X(m-i,n-5)\ \ldots\ X(m-i,n-N+1)]^T \quad (13)$$

$$A_{(i,2)} = [X(m-i,n-2)\ X(m-i,n-4)\ X(m-i,n-6)\ \ldots\ X(m-i,n-N)]^T \quad (14)$$

and

$$W_{(i,0)}(j-1) = [H_{j-1}(i,0)\ H_{j-1}(i,2)\ \ldots\ H_{j-1}(i,N-2)]^T \quad (15)$$

$$W_{(i,1)}(j-1) = [H_{j-1}(i,1)\ H_{j-1}(i,3)\ \ldots\ H_{j-1}(i,N-1)]^T \quad (16)$$

We can easily notice from equation (17), which we have derived directly from equation (11), that the filtering expression $A_{(i,1)}\,(W_{(i,0)}(j-1) + W_{(i,1)}(j-1))$ is repeated twice, so we should compute it only once. Therefore, the arithmetic complexity of the 2-D convolution is reduced from N^2 to $3N^2/4$. The same technique can be applied to the weight updating equations. After some manipulations we can get equation (18). Here also the expression $A_{(i,1)}\,(e(j-1) + e(j))$ is repeated twice and should be computed only once reducing, thereby, the arithmetic complexity from N^2 to $3N^2/4$. Moreover, the scalar value S(m,n) can be computed recursively (equation (19)), which reduces its arithmetic complexity from N^2 to 2N. For a block of 2 samples the arithmetic complexity of the proposed algorithm reduces from $2N^2$ to $3N^2/4 + 2N$. It is also possible to reduce the number of additions required by this algorithm. To that end, $(A_{(i,1)} - A_{(i,0)})$ and $(A_{(i,2)} - A_{(i,1)})$ should be computed iteratively from their results at the previous iteration.

3 THE FETDLMS ALGORITHM WITH A LARGER BLOCK SIZE

It is quite easy to extend the method explained above to larger block sizes. In the following, we consider, for the sake of illustration, only a block of size 4X4, and we suppose N to be a power-of-2 number. In this case, after modifying appropriately the different equations, equation (11) and (18) become equation (20) and (21), respectively. If now we subdivide each matrix in the above two equations into 4 subblocks each, then we can apply the fast filtering technique discussed above twice. This procedure gives, in fact, the matrix form of the nested radix-2 algorithm [3]. We can this way reduce the computational complexity of each equation from $(1/4)N^2$ to $(1/4)(3/4)^k N^2$ (per output) where k is the number of levels in the radix-2 algorithm. Fast filtering techniques with higher-radix exist [6], and can be used as well. They can reduce the computational complexity more than do radix-2 fast filtering algorithms. For the 2-D block LMS, the computational complexity depends on the technique used to achieve high-speed implementation of the convolution. This technique can be based, for example, on FFT or fast processors [4].

$$\begin{bmatrix} x_H^T(m-i,n-1) \\ x_H^T(m-i,n) \end{bmatrix} h_{j-1}(i) = \begin{bmatrix} A_{(i,1)}\,(W_{(i,0)}(j-1) + W_{(i,1)}(j-1)) + W_{(i,1)}(j-1)(A_{(i,2)} - A_{(i,1)}) \\ A_{(i,1)}\,(W_{(i,0)}(j-1) + W_{(i,1)}(j-1)) + W_{(i,0)}(j-1)(A_{(i,1)} - A_{(i,0)}) \end{bmatrix} \quad (17)$$

$$\begin{bmatrix} W_{(i,0)}(j+1) \\ W_{(i,1)}(j+1) \end{bmatrix} = \begin{bmatrix} W_{(i,0)}(j-1) \\ W_{(i,1)}(j-1) \end{bmatrix} + 2\mu \begin{bmatrix} A_{(i,1)}\,(e(j-1) + e(j)) - e(j)(A_{(i,1)} - A_{(i,0)}) \\ A_{(i,1)}\,(e(j-1) + e(j)) + e(j-1)(A_{(i,2)} - A_{(i,1)}) \end{bmatrix} \quad (18)$$

$$S(m,n)=S(m,n-2)+2\mu\sum_{i=0}^{N-1}\Big(X(m-l,n-1)\big(X(m-l,n)+X(m-l,n-2)\big)-X(m-l,n-N-1)\big(X(m-l,n-N)+X(m-l,n-N-2)\big)\Big) \quad (19)$$

$$\begin{bmatrix} x_H^T(m-i,n-3) \\ x_H^T(m-i,n-2) \\ x_H^T(m-i,n-1) \\ x_H^T(m-i,n) \end{bmatrix} h_{j-1}(i) = \begin{bmatrix} A(i,3) & A(i,4) & A(i,5) & A(i,6) \\ A(i,2) & A(i,3) & A(i,4) & A(i,5) \\ A(i,1) & A(i,2) & A(i,3) & A(i,4) \\ A(i,0) & A(i,1) & A(i,2) & A(i,3) \end{bmatrix} \begin{bmatrix} W_{(i,0)}(j-1) \\ W_{(i,1)}(j-1) \\ W_{(i,2)}(j-1) \\ W_{(i,3)}(j-1) \end{bmatrix} \quad (20)$$

$$\begin{bmatrix} W_{(i,0)}(j+1) \\ W_{(i,1)}(j+1) \\ W_{(i,2)}(j+1) \\ W_{(i,3)}(j+1) \end{bmatrix} = \begin{bmatrix} W_{(i,0)}(j-3) \\ W_{(i,1)}(j-3) \\ W_{(i,2)}(j-3) \\ W_{(i,3)}(j-3) \end{bmatrix} + 2\mu \begin{bmatrix} A^T(i,3) & A^T(i,2) & A^T(i,1) & A^T(i,0) \\ A^T(i,4) & A^T(i,3) & A^T(i,2) & A^T(i,1) \\ A^T(i,5) & A^T(i,4) & A^T(i,3) & A^T(i,2) \\ A^T(i,6) & A^T(i,5) & A^T(i,4) & A^T(i,3) \end{bmatrix} \begin{bmatrix} e(j-3) \\ e(j-2) \\ e(j-1) \\ e(j) \end{bmatrix} \quad (21)$$

---

4 SIMULATION RESULTS

The experimental behavior of the FETDLMS algorithm with a constant convergence factor has been tested by using the 2-D identification system shown in figure 1. The primary input is a white 2-D Gaussian noise, the 2-D unknown system is simulated with an order 8 hadamard filter whose gain has been normalized to 1, and lena image is used as the additive image. It is clear from figure 2, which gives the squared error between the true and the estimated 2-D FIR filter coefficients, that the classical 2-D block LMS convergence degrades as the block size is increased from 4X4 to 8X8. The FETDLMS algorithm is shown not only to converge faster than the 2-D block LMS, but remains insensitive to the change of the block size from 4X4 to 8X8. Thereby, the denoised images as shown in figure 3, 4 and 5 show the superiority of the FETDLMS when the block size increases. For each algorithm a convenient convergence factor has been selected.

5 CONCLUSION

In this paper, a new two-Dimensional block LMS algorithm was presented. First, we illustrated this with a block of 2 samples and showed that we can reduce by about 25% the number of multiplications required by the algorithm. Then, we gave some hints on how to extend this algorithm to larger block sizes, and showed that the reduction in arithmetic complexity increases too. The convergence performance of the algorithm, which was shown through simulations, is not affected by the change in the block size. Still much better results could be expected from the FETDLMS algorithm if a time varying convergence factor is used.

6 ACKNOWLEDGEMENTS

The authors would like to thank the UNESCO for partially supporting one of the authors (S. Attallah).

7 REFERENCES

[1] M. M. Hadhoud and David W. Thomas, "The Two-Dimensional Adaptive LMS (TDLMS) Algorithm", *IEEE Trans. Circuits and Syst.*, Vol. CAS-35, No. 5, pp. 485-494, May 1988.

[2] J. Benesty and P. Duhamel, "A Fast Exact Least Mean Square Adaptive Algorithm", *IEEE Trans. Sig. Proc.*, Vol. 40, No. 12, pp. 2904-2920, Dec. 1992.

[3] Z. J. Mou and P. Duhamel, "Fast Filtering: Algorithms and Implementations", Signal Processing, vol. 13, Dec. 1987, pp. 377-384, North-Holland.

[4] W. B. Mikhael and S. M. Ghosh "Two-dimensional block Adaptive Filtering Algorithms," *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS 92)* (San Diego), pp. 1219-1222, May. 1992.

[5] B. Widrow and S. D. Stearns, Adaptive Signal Processing. Englewood Cliffs, Prentice-Hall, NJ, 1985.

[6] P. C. Ballat, A. Antoniou and S. Morgera, "Higher Radix Aperiodic-Convolution Algorithms," *IEEE Trans. on Acoust. Speech, and Sig. Proc.*, Vol. ASSP-34, No. 1, pp. 60-68, Feb. 1986.

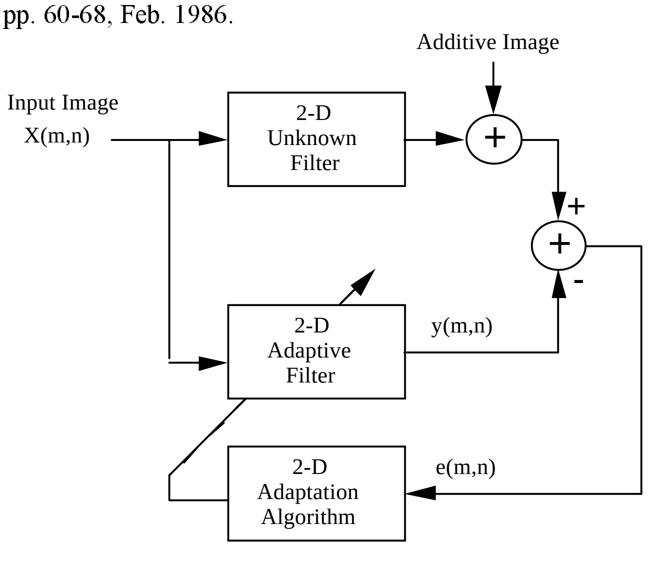

Fig. 1: 2-D System Identification

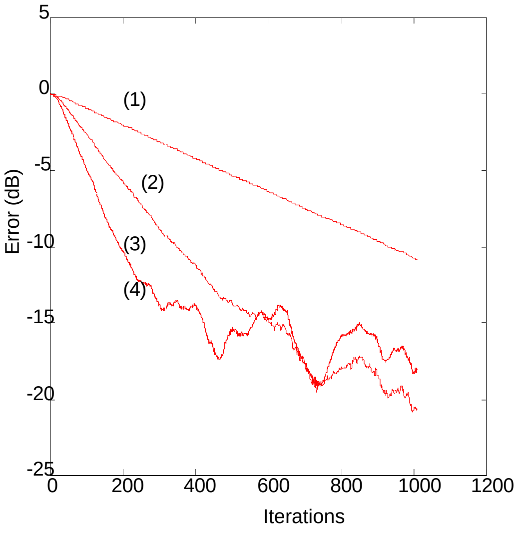

Figure 2: Comparison between the classical 2-D block LMS and the FETDLMS for different block sizes. (1): 2-D block LMS with a block size 8X8, (2): 2-D block LMS with a block size 4X4, (3): FETDLMS with a block size 8X8, and (4): FETDLMS with a block size 4X4

Figure 4: Image obtained with the 2-D block LMS with a block size 8X8

Figure 3: Image obtained with the 2-D block LMS with a block size 4X4

Figure 5: Image obtained with the FETDLMS with a block size 8X8