Multidelay Block Frequency Domain Adaptive Filter

JIA-SIEN SOO AND KHEE K. PANG

Abstract—**A flexible multidelay block frequency domain (MDF) adaptive filter is presented. The distinct feature of the MDF adaptive filter is to allow one to choose the size of an FFT tailored to the efficient use of a hardware, rather than the requirement of a specific application. The MDF adaptive filter also requires less memory and so reduces the requirement and cost of a hardware. In performance, the MDF adaptive filter introduces smaller block delay and is faster, ideal for a time-varying system such as modeling an acoustic path in a teleconference room. This is achieved by using smaller block size, updating the weight vectors more often, and reducing the total execution time of the adaptive process. The MDF adaptive filter compares favorably to other frequency domain adaptive filters when its adaptation speed and misadjustment are tested in computer simulations.**

I. Introduction

Adaptive digital filters have become increasingly popular due to their "intelligent" nature of processing signals and the emergence of a family of powerful digital signal processors. There are many adaptive algorithms available; each has its own merits and special applications. However, for applications such as an acoustic echo canceller in teleconference systems [1], which requires filter lengths of several hundreds and sometimes thousands, the frequency domain block adaptive filter based on the least mean-square algorithm (FLMS) [2] is considered to be most suitable. This is because the FLMS adaptive filter implements the block LMS (BLMS) [3] algorithm efficiently by using the fast Fourier transform (FFT). In so doing, a significant reduction in computational load for the same adaptation performance is achieved. Many other attractive features and variation of the FLMS adaptive filter can be found in [2]–[7]. However, a few practical implementation problems of the FLMS adaptive filter have hindered its applications. These are as follows.

1) Inefficient Use of a Hardware: For an adaptive filter, a $2N$-point FFT is generally used for an N-point weight factor. Most of the available FFT or DSP chips are designed and optimized for small size FFT, typically 256 point. To implement an acoustic echo canceller of a few thousands taps, several FFT chips are cascaded together with external memory to form a larger FFT configuration, which is rather inefficient and expensive.

2) Long Block Delay: Since the FLMS algorithm implements block processing, if the weight size $N = 1024$, the first output y_{k+1} needs to wait after the last output y_{k+1024} of the same block is processed or a delay of 128 ms for an 8 kHz sampling rate. Such a long delay would make the echo more annoying.

3) Large Quantization Error in FFT: As the size of an FFT becomes larger, the number of multiplications and scalings increases. This causes extra quantization error.

With these limitations in mind, we present a more flexible frequency domain adaptive filter structure, called the *multidelay block frequency domain* (MDF) adaptive filter in this correspondence. The performance of the MDF adaptive filter is compared to the existing frequency domain adaptive filters. It is found that by using a small FFT size and updating the weights more often, the MDF adaptive filter has a shorter block delay, faster adaptation speed, and small memory requirement.

Manuscript received February 9, 1988; revised May 19, 1989. This work was supported by the CSIRO (Australia) Collaborative Program in Information Technology.

The authors are with the Department of Electrical and Computer Systems Engineering, Monash University, Clayton, Victoria 3168, Australia.

IEEE Log Number 8932759.

Before we proceed to the next section, we denote the upper and lower case symbols as frequency and time domain variables, respectively. The boldface symbols will represent vectors or matrices. All vectors are specified as column vectors with superscript T to denote its transpose operation. An asterisk will denote complex conjugate transposition.

II. The MDF Adaptive Filters

To compute the linear convolution/correlation in the FLMS [2], [5] adaptive filter, either the overlap-save or overlap-add technique [4] is normally used. It is shown in [8] that by splitting the overlap-save method into two smaller blocks of an overlap-save process, the performance of the adaptive filter improves significantly. In this section, we extend the idea further to an arbitrary number of smaller delay blocks and generalize it to the MDF adaptive filter.

Let N be the total number of weights to be modeled and let M be the number of delay blocks. We then choose N' to be the size of the FFT, with N' equal to the smallest power of two integers larger than or equal to $2N/M$. The first step of the MDF algorithm is to convert the most recent overlapped input samples to the frequency domain via the FFT as

$$\begin{aligned}\boldsymbol{X}(M, j) = \operatorname{diag}\Big\{\mathrm{FFT}\big[&x_0(j-1), x_1(j-1), \\ &\cdots, x_{N'/2-1}(j-1), \\ &x_0(j), x_1(j), \cdots, x_{N'/2-1}(j)\big]^T\Big\} \end{aligned} \tag{1}$$

where j is the block iteration index. The earlier delay block input vectors are obtained via block index shifting without invoking any computation as follows:

$$\boldsymbol{X}(m, j) = \boldsymbol{X}(m+1, j-1), \qquad m = 1, 2, \cdots, M-1. \tag{2}$$

This suggests that only one FFT is needed per block iteration to transform the input vector, a significant computation saving. The output and error vectors can be expressed as

$$y(j) = \text{last } N'/2 \text{ terms of } \left\{\mathrm{FFT}^{-1}\left[\sum_{m=1}^{M} \boldsymbol{X}(m, j)\boldsymbol{W}(m, j)\right]\right\} \tag{3}$$

$$\boldsymbol{E}(j) = \mathrm{FFT}\Big\{\underbrace{0, 0, \cdots, 0}_{N'/2 \text{ zeros}}, \underbrace{[\boldsymbol{d}(j) - \boldsymbol{y}(j)]}_{N'/2 \text{ terms}}\Big\}^T \tag{4}$$

where $\boldsymbol{W}(m, j)$ is the mth weight vector and $\boldsymbol{d}(j)$ is the desired vector. The weight update equations based on the LMS criteria to minimize the $|\boldsymbol{e}(j)|^2$ are given as

$$\boldsymbol{\phi}(m, j) = \text{first half of } \left\{\mathrm{FFT}^{-1}[\boldsymbol{X}^*(m, j)\boldsymbol{E}(j)]\right\} \tag{5}$$

$$\boldsymbol{\Phi}(m, j) = \mathrm{FFT}\big[\boldsymbol{\phi}(m, j), \underbrace{0, 0, \cdots, 0}_{N'/2 \text{ zeros}}\big]^T \tag{6}$$

$$\boldsymbol{W}(m, j+1) = \boldsymbol{W}(m, j) + M\mu_B\boldsymbol{\Phi}(m, j) \tag{7}$$

where $m = 1, 2, \cdots, M$ and μ_B is the block step size.

The block diagram of the MDF adaptive filter depicted in Fig. 1 clearly illustrates the cascadable and efficient block structure. Note that the input/output operations of the MDF adaptive filter are identical to the FLMS, except now the FFT is N'-points long. The FLMS adaptive filter can, in fact, be regarded as the special case of MDF with $M = 1$. If the self-orthogonalizing algorithm is

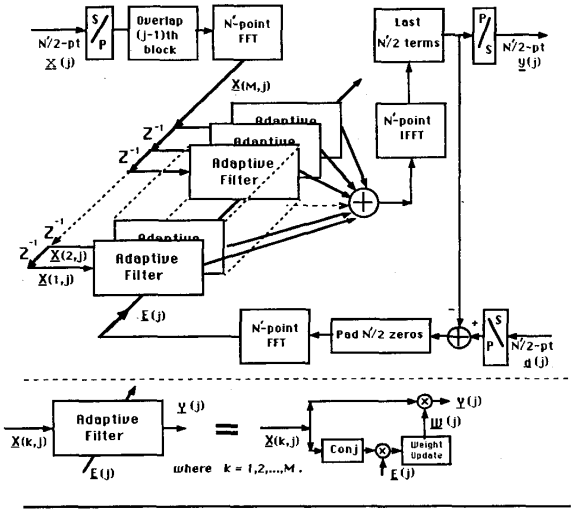

Fig. 1. A generalized MDF adaptive filter.

used, (7) becomes

$$\boldsymbol{W}(m, j+1) = \boldsymbol{W}(m, j) + \boldsymbol{\mu}(j)\boldsymbol{\Phi}(m, j) \tag{8}$$

$$\boldsymbol{\mu}(j) = \operatorname{diag}[\mu_0(j), \mu_1(j), \cdots, \mu_k(j), \cdots, \mu_{N'/2-1}(j)] \tag{9}$$

$$\mu_k(j) = M\mu_B / Z_k(j), \qquad k = 0, 1, \cdots, N'/2 - 1 \tag{10}$$

$$Z_k(j) = \beta Z_k(j-1) + (1-\beta)\left[\sum_{m=1}^{M} P_k(m, j)\right] \tag{11}$$

$$P_k(M, j) = X_k^*(M, j) X_k(M, j) \tag{12}$$

$$P_k(m, j) = P_k(m+1, j-1), \qquad m = 1, 2, \cdots, M-1 \tag{13}$$

where k is the frequency bin and β is the smoothing constant; $\beta = 0.8$ is used here.

It should be mentioned that the self-orthogonalizing algorithm applied here is not exactly equivalent to the algorithm proposed in [5] and [6]. The power estimate in (10)–(13) makes use of the Welch methods [10] to average the periodograms of each block. This results in the reduction of variance and smoothing of the power spectrum.

III. Simulation Results and Performance Analysis

In this section, we compare the performance of the MDF adaptive filter to other existing frequency domain adaptive filters. The simulation is based on identifying an FIR filter with the weights defined as

$$w_r' = (-1)^r \exp\left[-0.04(r+1)\right], \qquad r = 0, 1, \cdots, 127. \tag{14}$$

The input sample $x(j)$ is uniformly distributed between -100 and 100 and the quantization noise is added in the $d(j)$ by transforming the $d(j)$ from a real to an integer number. The normalized mean-square error (NMSE(k)) averaged over ten independent runs

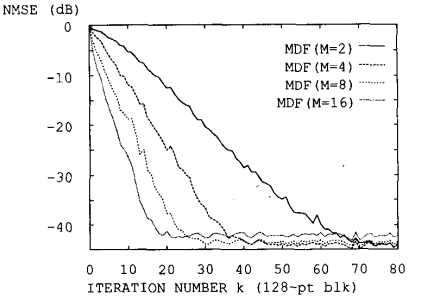

Fig. 2. Convergence characteristics of MDF adaptive filters for correlated input with eigenvalue ratio of 60 and self-orthogonalizing algorithm; $\mu_B = 0.1$, $Z_k(0) = 5.0E+5$, with $M = 2$, 4, 8, and 16.

in dB of the learning curve is defined as

$$\mathrm{NMSE}(k) = 10 \log \frac{\sum_{n=0}^{127} \left[d_n(k) - y_n(k)\right]^2}{\sum_{n=0}^{127} \left[d_n(k)\right]^2} \text{ dB}. \tag{15}$$

Fig. 2 illustrates that the convergence speed increases as the block size reduces when the step size μ_B is held constant. This observation agrees with the results in [3], in which the effect of block size in terms of time constant on the misadjustment is discussed. However, as the block size decreases, i.e., M increases, the time constant given does not predict the convergence speed accurately. In Fig. 2, the MDF adaptive filter at $M = 16$ is not twice as fast as that at $M = 8$. This discrepancy indicates that the time domain analysis for the BLMS algorithm used in [3] is not strictly applicable to its frequency domain counterpart.

For a realistic comparison of adaptation speed, we first choose the optimal step size to give the fastest convergence for the FLMS algorithm, then obtain two sets of step size for the MDF. The first set of step size is selected to give the same steady-state error as that of FLMS, whereas the second set (denoted by the asterisk) is obtained via trial and error to give the best performance. The MDF algorithm with both sets of step size is found to be faster than the FLMS algorithm in Fig. 3. The fast convergence rate is attributed to the more frequent weight update process adopted in the MDF algorithm. The above observation is consistent with the analysis found in [11].

To make the MDF adaptive filter more efficient, we explore the possibility of not imposing the weight constraint on (5), (6) as the UFLMS [5] does, and called the *unconstrained multidelay block frequency domain* (UMDF) adaptive filter. This results in a savings of two FFT operations for each delay block. However, the UMDF adaptive filter is found to be slower and has a larger misadjustment depending on the relative magnitude of the weights to be identified and the number of delay blocks. Alternatively, we can impose a weight constraint on only one block of weights at each block time j. That is, (5) and (6) are only implemented for only one block, but not all blocks of weight vectors at each iteration. In so doing, we are effectively time multiplexing or alternatively applying the weight constraint on each block. Thus, the name *alternative unconstrained multidelay block frequency domain* (AUMDF) adaptive filter is used. Fig. 4 shows that the AUMDF adaptive filter has almost identical performance compared to the MDF when M and μ_B are small. For $M = 16$ or μ_B large, there is some loss of performance for the AUMDF adaptive filter. Nevertheless, the choice of either an MDF, AUMDF, or UMDF adaptive filter is both application and hardware dependent.

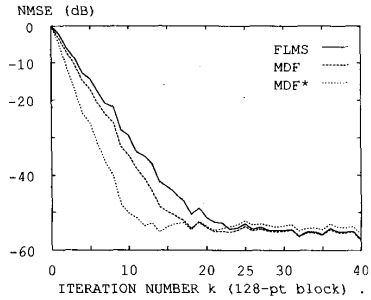

Fig. 3. Convergence characteristics of MDF ($M = 16$) and FLMS adaptive filters for uncorrelated input and step sizes: μ_B(MDF) = $0.625E - 7$, μ_B(MDF*) = $0.12E - 6$, and μ_B(FLMS) = $0.1E - 5$.

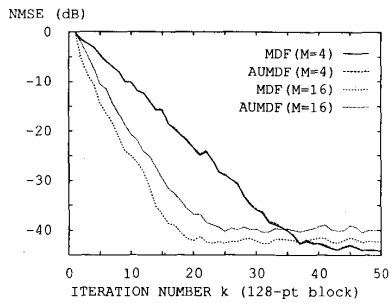

Fig. 4. Convergence characteristics of MDF and AUMDF adaptive filters. Same conditions as in Fig. 2.

TABLE I
EXECUTION TIME FOR THE ADAPTIVE FILTERS TO COMPUTE FFT

	Execution Time (in ms)						
	Complex FFT		Various Adaptive Filters				
DSP	256-Point	1024-Point	FLMS	UFLMS	MDF	AUMDF	UMDF
DSP56000	0.71	4.99	25.0	15.0	14.2	11.4	8.5
TMS320C25	1.22	7.10	35.5	21.3	24.4	19.5	14.6

IV. COMPUTATIONAL COMPLEXITY

Refer to Fig. 1; it is clearly shown that the number of multiplications of the frequency domain adaptive filter is directly proportional to the number and size of the FFT used. A $2N$-point FFT can be shown to require $N/2 \log_2 N$ complex multiplications or $2N \log_2 N$ real multiplications [9]. With this assumption, the total numbers of multiplications per output sample for different adaptive filters with fixed μ_B are

$$\text{FLMS: } 10 \log_2 N + 8 \tag{16}$$

$$\text{MDF: } (4M + 6) \log_2 N + 8M - (4M + 6) \log_2 M \tag{17}$$

$$\text{AUMDF: } 10 \log_2 N + 8M - 10 \log_2 M \tag{18}$$

$$\text{UMDF: } 6 \log_2 N + 8M - 6 \log_2 M. \tag{19}$$

The memory requirement of the proposed frequency domain adaptive filter is much less than the FLMS. For the MDF adaptive filter, the memory storage is approximately $4N + 10N/M$ points compared to roughly about $14N$ points for the FLMS algorithm. Thus, if $M \geq 4$, there is more than 50 percent reduction in memory storage for the MDF adaptive filter. The above comparison is based on the number of multiplications and memory storage required separately. To make a more practical comparison, we combine both of these two factors in terms of the total execution time needed to perform the FFT operations. We first assume $N = 512$ and the DSP used is either the Motorola DSP56000 or Ti TMS320C25. Since both of these DSP's are optimized for a 256-point FFT, we select $M = 4$ for the MDF adaptive filters. This means the size of the FFT required for the FLMS/UFLMS and the MDF adaptive filters are 1024 and 256 point, respectively. Table I summarizes the total execution time required for various adaptive filters. It is obvious that the MDF adaptive filters are more efficient in hardware utilization.

V. Conclusion

The MDF adaptive filter is derived and verified. It requires less memory storage, small FFT size, and allows different configurations to be chosen depending on the hardware used. In performance, the MDF adaptive filter has a smaller block delay and is faster. This is achieved by updating the weight vectors more often and reducing the total execution time in most of the DSP's. Furthermore, the total number of blocks needed can be changed dynamically without interrupting the normal operation. For example, one can add or drop one block of weight vector by checking the change in output error after some iterations to avoid the redundant operations. It is important to point out that the MDF adaptive filter is most suitable for real-time applications implemented on the DSP hardware, not on a general-purpose computer because the latter is not constrained by the small memory.

References

[1] N. Furuya, Y. Itoh, Y. Maruyama, and Araseki, "Audio conference equipment with acoustic echo canceller," *NEC Res. Develop.*, pp. 18–23, Jan. 1985.

[2] E. R. Ferrara, "Fast implementation of LMS adaptive filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-28, pp. 474–475, Aug. 1980.

[3] G. A. Clark, S. K. Mitra, and S. R. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 744–752, June 1981.

[4] G. A. Clark, S. R. Parker, and S. K. Mitra, "A unified approach to time- and frequency-domain realization of FIR adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 1073–1083, Oct. 1983.

[5] D. Mansour and A. H. Gray, Jr., "Unconstrained frequency-domain adaptive filter," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-30, pp. 726–734, Oct. 1982.

[6] E. R. Ferrara, Jr., "Frequency-domain adaptive filter," in *Adaptive Filters*, C. F. N. Cowan and P. M. Grant, Eds. Englewood Cliffs, NJ: Prentice-Hall, 1985, ch. 6.

[7] J. C. Lee and C. K. Un, "Performance of transform-domain LMS adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-34, pp. 499–510, June 1986.

[8] J. S. Soo and K. K. Pang, "A new structure for block FIR adaptive digital filters," in *IREECON Int. Dig. Papers*, Sydney, Australia, Sept. 1987, pp. 364–367.

[9] E. O. Brigham, *The Fast Fourier Transform*. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[10] P. D. Welch, "The use of fast Fourier transform for the estimation of power spectra," *IEEE Trans. Audio Electroacoust.*, vol. AU-15, pp. 70–73, June 1967.

[11] A. Feuer, "Performance analysis of the block least mean square algorithm," *IEEE Trans. Circuits Syst.*, vol. CAS-32, pp. 960–963, Sept. 1985.