Frequency Domain Equalization for High Data Rate Multipath Channels

Pawel A. Dmochowski and Peter J. McLane
Department of Electrical and Computer Engineering
Queen's University
Kingston, Ontario, Canada K7L 3N6

Abstract — **High data rate transmission over multipath channels requires equalizers of long impulse response. In such cases, frequency domain implementation of the block least mean square (BLMS) algorithm offers low complexity growth relative to time domain techniques. The work presented herein is devoted to a study of the fast BLMS (FBLMS) algorithm implemented in the frequency domain using overlap-save sectioning and the fast Fourier transform (FFT). We examine the bit error rate (BER) performance for high data rate quadrature phase shift keying (QPSK) transmission over a multipath channel as well as the computational complexity of the FBLMS equalizer in comparison to the time domain implementation. Finally, we show how normalizing the step size of the FBLMS algorithm according to the power distribution of the input process results in a significant improvement in the equalizer convergence relative to the time domain methods.**

I. Introduction

For time division multiple access (TDMA), the fundamental challenge in wireless transmission is overcoming intersymbol interference (ISI) caused by multipath propagation. The conventional method of mitigating ISI has been time domain equalization. However, as the signaling rate increases, the number of dispersed symbols grows linearly with it. As a result, the number of equalizer coefficients, N, needed to compensate for the ISI increases accordingly. Channels of interest in this paper include those for high data rate applications, where the intersymbol interference length is large. Since time domain algorithms have a computational complexity on the order of N^2, per block of N data symbols, such implementations may not be suitable for these applications even with today's state of digital signal processing technology.

Frequency domain equalization is a technique that offers low complexity growth with an increase of equalizer length in comparison to the time domain approach [1]. By efficiently implementing block data processing using overlap-save sectioning and the fast Fourier transform (FFT), the complexity can be reduced to the order of $NlogN$ per block of N data symbols. For a large number of equalizer taps, this translates to substantial computational savings. We present an analysis of the frequency domain fast block LMS (FBLMS) and normalized FBLMS (NFBLMS) algorithms as applied to static multipath channels with long delay spreads. Performance measures such as the bit error rate, computational complexity and the rate of convergence are examined and compared to the conventional time domain equalization approach.

II. Fast Block LMS Algorithm

A Block LMS Algorithm

The FBLMS algorithm is an efficient frequency domain implementation of the Block LMS (BLMS) algorithm. The BLMS algorithm is a generalized form of the LMS algorithm, in which the gradient is estimated over a block of L input samples. The resulting update equation for the equalizer tap column vector $\mathbf{w}(n)$ is given by [2]

$$\mathbf{w}(n+L) = \mathbf{w}(n) + 2\mu \sum_{i=0}^{L-1} \mathbf{x}(n+i)e^*(n+i), \qquad (1)$$

where μ is the adaptation step size and $\mathbf{x}(n)$ is the column vector of the equalizer input. The error, $e(n)$, is the difference between the desired response, $d(n)$, and the equalizer output, $y(n)$, that is

$$e(n+i) = d(n+i) - y(n+i) \qquad i = 0, \ldots, L-1. \qquad (2)$$

In the block implementation of the LMS algorithm, the equalizer taps are held constant for the duration of the data block, hence the equalizer output can be expressed as

$$y(n+i) = \mathbf{w}^H(n)\mathbf{x}(n+i) \qquad i = 0, \ldots, L-1, \qquad (3)$$

where the superscript H denotes a Hermitian transpose. The computation of the equalizer output in Equation (3) and the estimation of the gradient in Equation (1) are the operations of linear convolution and linear correlation, respectively. The basis of the frequency domain BLMS algorithm is an efficient implementation of these operations via the fast Fourier transform (FFT) [3].

The main challenge in implementing these operations in the frequency domain is that FFT operation leads to circular convolution/correlation [4]. In order to implement the FBLMS algorithm properly, proper data sectioning must be carried out to eliminate the wrap around effects of circular convolution/correlation. The most common data sectioning methods used are overlap-save and overlap-add. We describe the former, as it leads to a more efficient implementation of the FBLMS algorithm [3].

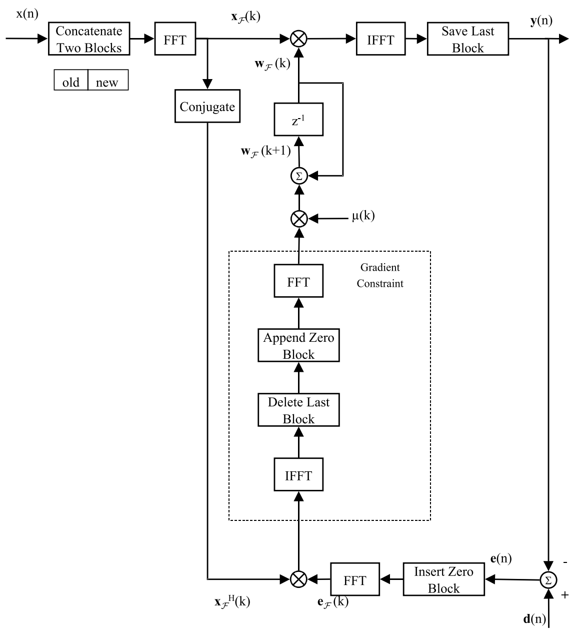

Fig. 1. The FBLMS Algorithm

B Overlap-Save Data Sectioning

In order to eliminate the aliasing effects of circular convolution, the lengths of the FFT's used must be sufficient. Specifically, to convolve a block of N symbols with a filter of length N, it is necessary to use FFT's of length $2N$ [4]. Overlap-save is a method of computing linear convolution of a long data sequence with filter taps by sectioning the input data stream into blocks, and obtaining the output blocks using the FFT. The input samples are broken down into blocks of length $2N$ with adjacent blocks overlapping by N samples. The filter tap vector is also extended by padding it with N zeros. The $2N$-point FFT's are computed and the output block is obtained from the inverse FFT of their product. Since the first N output symbols are the result of aliasing, only the last N samples constitute valid filter output. The above process is repeated for all data blocks, with the individual output blocks appended together to give the overall output sequence.

C FBLMS Algorithm

The FBLMS algorithm implemented using overlap save is shown in Figure 1. The input data stream is broken down into blocks of $2N$, consisting of N samples from the previous block followed by N new ones. The FFT of the resulting samples can be expressed as a diagonal matrix

$$\mathbf{X}_{\mathcal{F}}(k) = diag\{\mathcal{F}\{x(kN-N), ..., x(kN+N-1)\}\}, \quad (4)$$

where k is the block index and $\mathcal{F}$ refers to the Fourier transform operation. The FFT of the equalizer taps is a column vector of length $2N$, as the tap values are augmented at the end with N zeros, that is

$$\mathbf{w}_{\mathcal{F}}(k) = [\mathcal{F}\{w_o(n), ..., w_{N-1}(n), 0, ..., 0\}]^T, \quad (5)$$

where the superscript T denotes a transpose. The Fourier transform of the output vector is the element-by-element product of the transformed sequences, given by

$$\mathbf{y}_{\mathcal{F}}(k) = \mathbf{X}_{\mathcal{F}}(k)\mathbf{w}_{\mathcal{F}}(k). \quad (6)$$

As discussed previously, the first N samples of the product are invalid as they are the result of circular convolution. The vector $\mathbf{y}_{\mathcal{F}}(k)$ must be transformed into the time domain, where the invalid samples are discarded and only the last N are used to compute the error

$$e(kN+m) = d(kN+m) - y(kN+m) \qquad m = 0, ..., N-1. \quad (7)$$

The above constraint can be formulated using the following matrix notation. By defining a sectioning constraint matrix for the error block as

$$\mathbf{K} = [0_N \quad I_N], \quad (8)$$

where 0_N is a $N \times N$ zero matrix, I_N is an $N \times N$ identity matrix, the time domain output vector can be expressed as

$$\mathbf{y}(n) = \mathbf{K}\mathcal{F}^{-1}\mathbf{X}_{\mathcal{F}}(k)\mathbf{w}_{\mathcal{F}}(k), \quad (9)$$

where $\mathcal{F}$ is a $2N \times 2N$ DFT matrix with elements $\mathcal{F}_{ij} = e^{-j2\pi ij/2N}$. The resulting error samples are augmented with N zeros,

$$\mathbf{e}_{\mathcal{F}}(k) = [\mathcal{F}\{0, ..., 0, e(kN), ..., e(kN+N-1)\}]^T, \quad (10)$$

since linear correlation has to be computed between the error vector and the complex conjugate of the input vector $\mathbf{X}_{\mathcal{F}}^{H}(k)$. Using the constraint matrix, the vector $\mathbf{e}_{\mathcal{F}}(k)$ can be rewritten as

$$\mathbf{e}_{\mathcal{F}}(k) = \mathcal{F}\mathbf{K}^T\mathbf{e}(k). \quad (11)$$

The next section of the algorithm is the gradient constraint. Since there are only N tap weights being updated in the time domain, it is necessary to ensure that the gradient contains only N non-zero elements. Thus, the gradient is transformed into the time domain and the last N elements are replaced with zeros before transforming it back to frequency domain and updating the tap values. The gradient constraint can be expressed by using a constraint matrix similar to that used for the error block. By defining $\mathbf{P}$ as

$$\mathbf{P} = \begin{bmatrix} I_N & 0_N \\ 0_N & 0_N \end{bmatrix}, \quad (12)$$

the augmented gradient can be expressed as

$$[\hat{\nabla}\mathbf{J}(k), 0, ..., 0]^T = \mathbf{P}\mathcal{F}^{-1}\mathbf{X}_{\mathcal{F}}^{H}(k)\mathbf{e}_{\mathcal{F}}(k), \quad (13)$$

where $\hat{\nabla}\mathbf{J}(k)$ refers to the estimated gradient. The tap update therefore becomes

$$\mathbf{w}_{\mathcal{F}}(k+1) = \mathbf{w}_{\mathcal{F}}(k) + 2\mu\mathcal{F}\mathbf{P}\mathcal{F}^{-1}\mathbf{X}_{\mathcal{F}}^{H}(k)\mathbf{e}_{\mathcal{F}}(k). \quad (14)$$

D Normalized FBLMS

The convergence properties of the FBLMS algorithm can be greatly improved by normalizing the step size inversely proportional to the power content of each frequency bin [2]. This is equivalent to replacing the step size μ with $\mu\mathbf{D}^{-1}$, where $\mathbf{D}$ is the diagonal matrix with the detected variance of the input process, $\sigma^2_{x_{\mathcal{F},i}}(n)$, as its entries. In practice, the matrix $\mathbf{D}$ is replaced by $\hat{\mathbf{D}}$, with estimates $\hat{\sigma}^2_{x_{\mathcal{F},i}}(n)$ on its main diagonal. Each entry in $\hat{\mathbf{D}}$ is computed by a recursion formula [2]

$$\hat{\sigma}^2_{x_{\mathcal{F},i}}(n) = \gamma\hat{\sigma}^2_{x_{\mathcal{F},i}}(n-1) + (1-\gamma)\sigma^2_{x_{\mathcal{F},i}}(n), i = 0, \ldots, N-1, \quad (15)$$

where γ is often referred to as the *smoothing factor*. The resulting algorithm, referred to as the normalized FBLMS (NFBLMS), has the tap update equation given by

$$\mathbf{w}_{\mathcal{F}}(k+1) = \mathbf{w}_{\mathcal{F}}(k) + 2\mu\hat{\mathbf{D}}^{-1}\mathcal{F}\mathbf{P}\mathcal{F}^{-1}\mathbf{X}^H_{\mathcal{F}}(k)\mathbf{e}_{\mathcal{F}}(k). \quad (16)$$

III. Simulation Methodology

The communication system model, depicted in Figure 2, consisted of a cascade of a data source, transmit filter, channel, receive filter, equalizer, a decision device and a data sink. The pulse shaping was performed by a square root raised cosine transmit filter with a roll-off of 35%. Since the channel characteristics were assumed to be unknown, we employed a suboptimum receive filter, that is one matched only to the transmit filter and not the combination of the transmitter and the channel. Finally, a T-spaced linear equalizer was used following the receiver.

The simulation results presented here were conducted on a multipath channel [1] with a maximum delay spread of $8\mu s$ [5]. Its impulse response was obtained by sampling the delay profile at 8 times the symbol rate and varying the phase for each symbol according to a uniform distribution from 0 to 2π. Defining the bandwidth at the frequency of a 5 dB amplitude drop, the channel has a bandwidth between 150 kHz and 250 kHz, depending on the phase characteristics. The input signal-to-interference ratio (SIR_{input}) of the channel, expressed as

$$SIR_{input} = \frac{f_0 f_0^*}{\sum\limits_{i\neq 0} f_i f_i^*}, \quad (17)$$

where f_i denotes the impulse response of the overall (unequalized) channel, varies between 5 dB and -2 dB between data rates of 1.0 Mbps to 4.0 Mbps.

IV. Bit Error Rate Performance

The first performance criterion examined is the bit error rate (BER). In [6], Beaulieu developed an efficient analytical method for calculating the bit error rate over channels with both cross-talk and ISI and corrupted by additive Gaussian

[1]Other channels considered can be found in [5].

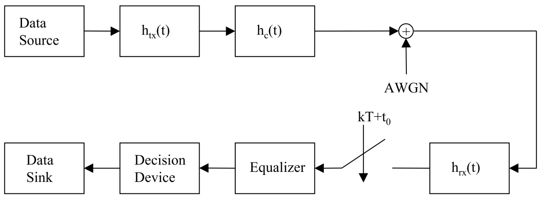

Fig. 2. Communication System Model

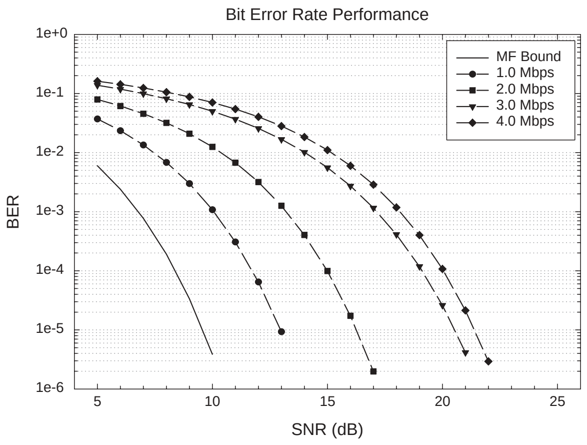

Fig. 3. Bit Error Rate Performance

noise. This method, which is based on the Fourier series expansion of the noise cumulative distribution function (cdf), was used throughout this paper.

We examine the BER performance as a function of the signal-to-noise ratio E_b/N_0, where N_0 is the noise variance and E_b is the energy per data bit. Figure 3 shows the results for QPSK signaling at data rates from 1.0 Mbps to 4.0 Mbps. The matched filter bound - the performance achievable assuming transmission of isolated symbols, is also included as a reference point. The equalizers used consisted of 32 symbol-spaced taps for signaling rates up to 2.0 Mbps and 64 taps for transmission rates of over 2.0 Mbps. Since, providing a gradient constraint is used, the frequency domain algorithm yields equalizer taps identical to those obtained by the time domain methods, their bit error rate performance is identical.

From Figure 3 we see that an SNR of approximately 11.5 dB is required to achieve BER of 10^{-4} for 1.0 Mbps transmission. In order to maintain a steady BER, the signal power must be increased by approximately 4 dB for each 1.0 Mbps increase in transmission rate between 1.0 and 3.0 Mbps. The performance drop between 3.0 and 4.0 Mbps is approximately 1.5 dB.

V. Computational Complexity

Next, we compare the computational complexity for the time and frequency domain algorithms. For QPSK signaling, the total number of real multiplications used by the

LMS, FBLMS and NFBLMS algorithms for a block of N symbols are $8N^2$, $20Nlog2N + 16N$ and $20Nlog2N + 20N$ respectively [5]. The results are tabulated in Table 1, which lists the number of real multiplications needed for a block of N symbols for N equal to 32 and 64. In the calculations we have assumed QPSK transmission and complex equalizer taps.

Table 1: Equalizer Complexity Comparison

	$N = 32$	$N = 64$
LMS	8192	32768
FBLMS	4352	9984
NFBLMS	4480	10240

We see that both the normalized and non-normalized frequency domain block LMS algorithms offer significant savings in computational complexity over the LMS counterpart. Gains of 50 % and 65% are obtained for block lengths of 32 and 64. As stated in [1], current DSP's are capable of performing up to 1.6×10^9 real operations per second. Assuming a transmission rate of 2×10^6 complex symbols per second, the DSP will be capable of performing approximately 800 real operations per complex symbol. Taking into account that the values given in Table 1 are for blocks of up to 64 complex symbols, 51200 real operations per block can be provided by the DSP. Since, at most 10240 operations are needed, modern DSP's should be well suitable for frequency domain equalization applications.

VI. Convergence Properties

Finally, we analyze the convergence rate of the algorithms. We present results for 4 Mbps QPSK transmission over the channel previously described with an equalizer of length 64. Figure 4 shows plots of the mean square error as a function of the iteration number for an average of 100 independent trials. The NFBLMS algorithm exhibits a much greater convergence rate compared to the LMS and FBLMS implementations. The NFBLMS algorithm converged to a steady state after about 25 data blocks of 64 symbols, while the LMS and FBLMS algorithms failed to converge after processing over 50 data blocks. One must note, however, that while the normalized FBLMS algorithm exhibits a large improvement in the learning curve, the convergence rate still remains prohibitively slow for most applications.

VII. Conclusion

This work presented a study of an efficient, frequency domain implementation of the Block LMS algorithm. Using the overlap save sectioning and an FFT method of convolution, an algorithm with significantly improved complexity was applied to a high rate multipath channel. The BER results were presented for signaling rates of 1.0 to 4.0 Mbps QPSK. Computational complexity and convergence properties were compared for LMS, FBLMS and NFBLMS algorithms. While the computational complexity and the rate of

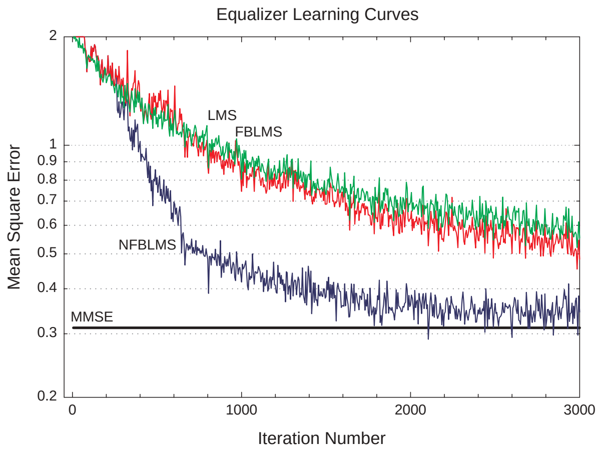

Fig. 4. Equalizer Learning Curves

convergence were significantly improved, the latter remains slow. For most practical applications, one might consider implementing an efficient version of a fast converging algorithm, such as the RLS algorithm.

VIII. Acknowledgement

The authors are grateful to the Natural Sciences and Engineering Research Council of Canada for their generous financial support.

References

[1] M. V. Clark. Adaptive frequency-domain equalization and diversity combining for broadband wireless communications. *IEEE Journal on Selected Areas in Communications*, 16(8):1385–1395, October 1998.

[2] B. Farhang-Boroujeny. *Adaptive Filters: Theory and Applications*. John Wiley & Sons Ltd., Toronto, ON, 1998.

[3] J. J. Shynk. Frequency-domain and multirate adaptive filtering. *IEEE Signal Processing Magazine*, 9(1):14–37, January 1992.

[4] A. V. Oppenheim and R. W. Schafer. *Discrete-Time Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[5] P. A. Dmochowski. Frequency domain equalization for high data rate multipath channels. Master's thesis, Queen's University, 2001.

[6] N. C. Beaulieu. The evaluation of error probabilities for intersymbol and cochannel interference. *IEEE Transactions on Communications*, 39(12):1740–1749, December 1991.