TDA Progress Report 42-121 May 15, 1995

An Efficient Implementation of Forward–Backward Least-Mean-Square Adaptive Line Enhancers

H.-G. Yeh
Spacecraft Telecommunications Equipment Section

T. M. Nguyen
Communications Systems Research Section

An efficient implementation of the forward-backward least-mean-square (FBLMS) adaptive line enhancer is presented in this article. Without changing the characteristics of the FBLMS adaptive line enhancer, the proposed implementation technique reduces multiplications by 25 percent and additions by 12.5 percent in two successive time samples in comparison with those operations of direct implementation in both prediction and weight control. The proposed FBLMS architecture and algorithm can be applied to digital receivers for enhancing signal-to-noise ratio to allow fast carrier acquisition and tracking in both stationary and nonstationary environments.

I. Introduction

Adaptive line enhancers (ALEs) are useful in many areas, including time-domain spectral estimation for fast carrier acquisition [2–4]. For example, a fast carrier acquisition technique [2],[1] as shown in Fig. 1, will be very useful for a deep-space mission, especially in a nonstationary environment or emergencies. Figure 1 is the block diagram of an ALE in a digital receiver used for both acquisition and tracking. First, the receiver is in the acquisition mode. Second, when the uplink carrier is acquired as indicated by the lock detector, the switch is shifted to the tracking position and the tracking process takes over immediately. With this acquisition scheme, the uplink carrier can be acquired by a transponder in seconds (as opposed to minutes for the Cassini transponder). Although devised to support a space mission, the architecture of the forward–backward least-mean-square (FBLMS) ALE and the associated algorithm proposed in this article are also applicable to other systems, including fixed-ground and mobile communication systems. Note that this proposed ALE scheme in the receiver needs a residual carrier, and does not work directly in suppressed-carrier cases.

A conventional ALE system using a least-mean-square (LMS) algorithm is depicted in Fig. 2, where z^{-1} represents a delay. The analysis of the ALE for enhancing the signal-to-noise ratio (SNR) to allow fast acquisition is given in [2]. The block diagram of a FBLMS adaptive line enhancer is shown in Fig. 3. The performance analysis of the FBLMS adaptive line enhancer is provided in [1]. The FBLMS adaptive line enhancer algorithm enjoys approximately half the misadjustment of that of the LMS algorithm [1].

[1] T. M. Nguyen, H. G. Yeh, and L. V. Lam, "A New Carrier Frequency Acquisition Technique for Future Digital Transponders," to be published in a future issue of *The Telecommunications and Data Acquisition Progress Report.*

References

[1] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal*, vol. 27, pp. 379–423 and 623–656, 1948.

[2] S. J. Dolinar and F. Pollara, "The Theoretical Limits of Source and Channel Coding," *The Telecommunications and Data Acquisition Progress Report 42-102, April–June 1990*, Jet Propulsion Laboratory, Pasadena, California, pp. 62–72, August 15, 1990.

[3] S. A. Butman and R. J. McEliece, "The Ultimate Limits of Binary Coding for a Wideband Gaussian Channel," *The Deep Space Network Progress Report 42-22, May–June 1974*, Jet Propulsion Laboratory, Pasadena, California, pp. 78–80, August 15, 1974.

$$e_f(n) = x(n) - \mathbf{X}^T(n)\mathbf{W}(n) \tag{1a}$$

$$e_b(n) = x(n-N) - \mathbf{X}_b^T(n)\mathbf{W}(n) \tag{1b}$$

where the superscript T denotes the transpose of a vector, and

$$\mathbf{X}^T(n) = [x(n-1), x(n-2), \cdots, x(n-N)] \tag{1c}$$

$$\mathbf{X}_b^T(n) = [x(n-N+1), x(n-N+2), \cdots, x(n)] \tag{1d}$$

$$\mathbf{W}^T(n) = [w_1(n), w_2(n), \cdots, w_N(n)] \tag{1e}$$

In any gradient algorithm, the coefficient vector $\mathbf{W}(n)$ is updated using

$$\mathbf{W}(n+1) = \mathbf{W}(n) - \mu\hat{\nabla}\{e(n)^2\} \tag{2a}$$

where μ is the adaptive step size and the $\hat{\nabla}\{e(n)^2\}$ is the estimated gradient of the surface of $E\{e(n)^2\}$. Note that $E\{\cdot\}$ denotes the expected value. In the forward–backward algorithm, $e(n)^2 = e_f(n)^2 + e_b(n)^2$, and the gradient estimate is chosen as

$$\hat{\nabla}\{e(n)^2\} = -[e_f(n)\mathbf{X}(n) + e_b(n)\mathbf{X}_b(n)] \tag{2b}$$

It is shown in [1] that Eq. (2b) is an unbiased estimator of the gradient. This leads to the coefficient update

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu[e_f(n)\mathbf{X}(n) + e_b(n)\mathbf{X}_b(n)] \tag{2c}$$

This means that $\mathbf{W}(n+1) \cong \mathbf{W}(n)$ in steady state when both forward and backward errors are approaching zero.

III. The Fast Forward–Backward LMS Algorithm

The fast FBLMS algorithm is derived in this section by using the radix-2 algorithm on time samples. Both predictor and weight update sections are provided in detail.

A. Predictor Section

We consider the computation of two successive predictions in both forward and backward directions with the fixed weight coefficient $\mathbf{W}(n-1)$. After regrouping even and odd terms, the forward predictor is obtained [5] and given in Eq. (3):

$$\begin{bmatrix} \hat{d}_f(n-1) \\ \hat{d}_f(n) \end{bmatrix} = \begin{bmatrix} \mathbf{X}^T(n-1) \\ \mathbf{X}^T(n) \end{bmatrix} \mathbf{W}(n-1) = \begin{bmatrix} \mathbf{A}^T & \mathbf{B}^T \\ \mathbf{C}^T & \mathbf{A}^T \end{bmatrix}_n \begin{bmatrix} \mathbf{W}_0 \\ \mathbf{W}_1 \end{bmatrix}_{n-1} \tag{3a}$$

where

$$\mathbf{A}^T = [x(n-2), x(n-4), \cdots, x(n-N+2), x(n-N)] \tag{3b}$$

$$\mathbf{B}^T = [x(n-3), x(n-5), \cdots, x(n-N+1), x(n-N-1)] \tag{3c}$$

$$\mathbf{C}^T = [x(n-1), x(n-3), \cdots, x(n-N+3), x(n-N+1)] \tag{3d}$$

$$\mathbf{W}_0 = [w_0(n-1), w_2(n-1), \cdots, w_{N-2}(n-1)]^T \tag{3e}$$

$$\mathbf{W}_1 = [w_1(n-1), w_3(n-1), \cdots, w_{N-1}(n-1)]^T \tag{3f}$$

Similarly, the backward predictor is obtained and given as follows:

$$\begin{bmatrix} \hat{d}_b(n-1) \\ \hat{d}_b(n) \end{bmatrix} = \begin{bmatrix} \mathbf{X}_b^T(n-1) \\ \mathbf{X}_b^T(n) \end{bmatrix} \mathbf{W}(n-1) = \begin{bmatrix} \mathbf{F}^T & \mathbf{G}^T \\ \mathbf{G}^T & \mathbf{H}^T \end{bmatrix}_n \begin{bmatrix} \mathbf{W}_0 \\ \mathbf{W}_1 \end{bmatrix}_{n-1} \tag{4a}$$

where

$$\mathbf{F}^T = [x(n-N), x(n-N+2), \cdots, x(n-4), x(n-2)] \tag{4b}$$

$$\mathbf{G}^T = [x(n-N+1), x(n-N+3), \cdots, x(n-3), x(n-1)] \tag{4c}$$

$$\mathbf{H}^T = [x(n-N+2), x(n-N+4), \cdots, x(n-2), x(n)] \tag{4d}$$

Equations (3a) and (4a) are approximations by virtue of updating the weight vector only once every two cycles. The relationship between the two sequence sets $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ and $\{\mathbf{F}, \mathbf{G}, \mathbf{H}\}$ is given as follows:

$$\mathbf{F} = \mathbf{A}_r \tag{5}$$

$$\mathbf{G} = \mathbf{C}_r \tag{6}$$

$$z^{-1}\mathbf{H} = \mathbf{A}_r \tag{7}$$

where subscript r means the reversed order of the sequence and the z^{-1} means one delay unit of the corresponding sequence and is equivalent to two time sample delays. Furthermore, we observe the following relationships between **G**, **B**, **C**:

$$z^{-1}\mathbf{G} = \mathbf{B}_r \tag{8}$$

$$z^{-1}\mathbf{C} = \mathbf{B} \tag{9}$$

After performing the appropriate computation, Eq. (4a) can be rewritten as follows:

$$\begin{bmatrix} \hat{d}_b(n-1) \\ \hat{d}_b(n) \end{bmatrix} = \begin{bmatrix} \mathbf{G}^T(\mathbf{W}_0 + \mathbf{W}_1) + (\mathbf{F} - \mathbf{G})^T \mathbf{W}_0 \\ \mathbf{G}^T(\mathbf{W}_0 + \mathbf{W}_1) - (\mathbf{G} - \mathbf{H})^T \mathbf{W}_1 \end{bmatrix} \tag{10}$$

The computation of Eq. (4a) requires two inner products of length N, while that of Eq. (10) requires only three inner products of length $N/2$ and $N/2$ additions to perform $\mathbf{W}_0 + \mathbf{W}_1$. Similarly, by combining Eqs. (5) through (9), Eq. (3a) can be rewritten as follows:

$$\begin{aligned} \begin{bmatrix} \hat{d}_f(n-1) \\ \hat{d}_f(n) \end{bmatrix} &= \begin{bmatrix} \mathbf{A}^T(\mathbf{W}_0 + \mathbf{W}_1) + (\mathbf{B} - \mathbf{A})^T \mathbf{W}_1 \\ \mathbf{A}^T(\mathbf{W}_0 + \mathbf{W}_1) - (\mathbf{A} - \mathbf{C})^T \mathbf{W}_0 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}^T(\mathbf{W}_0 + \mathbf{W}_1) + z^{-1}(\mathbf{G} - \mathbf{H})_r^T \mathbf{W}_1 \\ \mathbf{A}^T(\mathbf{W}_0 + \mathbf{W}_1) - (\mathbf{F} - \mathbf{G})_r^T \mathbf{W}_0 \end{bmatrix} \end{aligned} \tag{11}$$

Clearly, the sequences $(\mathbf{G} - \mathbf{H})$ and $(\mathbf{F} - \mathbf{G})$ of Eq. (10) are reused again in Eq. (11), but in reverse order. The computation of Eq. (11) requires only three inner products of length $N/2$. The total number of multiplications and additions required in both forward and backward predictor sections for two successive computations is about $3N$ and $3.5N$, respectively. The total number of multiplications and additions required in Eqs. (1a) and (1b) for two successive prediction sections is $4N$ and $4(N-1)$. Consequently, there are about 25 percent and 12.5 percent savings in multiplications and additions, respectively.

B. Weight Update Section

We consider the weight coefficient updates now. Since weights are explicitly computed at every other time update using the look-ahead approach [6], the weight update of Eq. (2c) can be rewritten as follows:

$$\begin{aligned} \mathbf{W}(n+1) &= \mathbf{W}(n-1) + \mu\left[e_f(n-1)\mathbf{X}(n-1) + e_b(n-1)\mathbf{X}_b(n-1)\right] + \mu\left[e_f(n)\mathbf{X}(n) + e_b(n)\mathbf{X}_b(n)\right] \\ &= \mathbf{W}(n-1) + [\mathbf{X}(n) \quad \mathbf{X}(n-1)] \begin{bmatrix} \mu e_f(n) \\ \mu e_f(n-1) \end{bmatrix} + [\mathbf{X}_b(n) \quad \mathbf{X}_b(n-1)] \begin{bmatrix} \mu e_b(n) \\ \mu e_b(n-1) \end{bmatrix} \end{aligned} \tag{12}$$

By combining Eqs. (5) through (9), Eq. (12) is rewritten as follows:

$$\begin{aligned} \begin{bmatrix} \mathbf{W}_0 \\ \mathbf{W}_1 \end{bmatrix}_{n+1} &= \begin{bmatrix} \mathbf{W}_0 \\ \mathbf{W}_1 \end{bmatrix}_{n-1} + \begin{bmatrix} \mathbf{C} & \mathbf{A} \\ \mathbf{A} & \mathbf{B} \end{bmatrix} \begin{bmatrix} \mu e_f(n) \\ \mu e_f(n-1) \end{bmatrix} + \begin{bmatrix} \mathbf{G} & \mathbf{F} \\ \mathbf{H} & \mathbf{G} \end{bmatrix} \begin{bmatrix} \mu e_b(n) \\ \mu e_b(n-1) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{W}_0 \\ \mathbf{W}_1 \end{bmatrix}_{n-1} + \mu \begin{bmatrix} \mathbf{A}(e_f(n) + e_f(n-1)) - (\mathbf{F} - \mathbf{G})_r e_f(n) \\ \mathbf{A}(e_f(n) + e_f(n-1)) + z^{-1}(\mathbf{G} - \mathbf{H})_r e_f(n-1) \end{bmatrix} \\ &\quad + \mu \begin{bmatrix} \mathbf{G}(e_b(n) + e_b(n-1)) + (\mathbf{F} - \mathbf{G}) e_b(n-1) \\ \mathbf{G}(e_b(n) + e_b(n-1)) - (\mathbf{G} - \mathbf{H}) e_b(n) \end{bmatrix} \end{aligned} \tag{13}$$

The vectors $(\mathbf{F} - \mathbf{G})$ and $(\mathbf{G} - \mathbf{H})$ are once more employed in Eq. (13). Notice that the term $\mu[\mathbf{A}(e_f(n) + e_f(n-1)) + \mathbf{G}(e_b(n) + e_b(n-1))]$ is computed only once, and the sum is applied to both $\mathbf{W}_0$ and $\mathbf{W}_1$ for updates. The total numbers of multiplications and additions in Eq. (13) are about $3N$ and $3.5N$, respectively. However, the total numbers of multiplications and additions of Eq. (2c) for two adaptations are $4N$ and $4(N-1)$. Consequently, 25 percent of multiplications and 12.5 percent of additions are saved by using Eq. (13) in comparison with those operations of Eq. (2c).

IV. Implementation

The architecture of the fast FBLMS algorithm is depicted in Fig. 4. A switching circuit is employed after the adaptive line enhancer, and the switch rate (from $\mathbf{C}$ to $\mathbf{A}$ or from $\mathbf{A}$ to $\mathbf{C}$) is the same as the sampling rate. The switching circuit is switched between points $\mathbf{C}$ and $\mathbf{A}$ alternately. Sequences $\mathbf{C}$ and $\mathbf{A}$ are generated at a rate of $1/(2T)$ accordingly. The sequence $\mathbf{B}$ is a delayed version of the sequence $\mathbf{C}$. By using a radix-2 structure, sequences $\{\mathbf{B} - \mathbf{A}\}$ and $\{\mathbf{A} - \mathbf{C}\}$ are then generated at the upper and lower lag, respectively. By using the sequence $\{\mathbf{B} - \mathbf{A}\}$, inner products $(\mathbf{B} - \mathbf{A})^T\mathbf{W}_1$ and $z^{-1}(\mathbf{G} - \mathbf{H})^T\mathbf{W}_1$ are generated at the upper and lower lag, respectively, of the upper forward–backward tapped-delay-line structure. Similarly, by using the sequence $\{\mathbf{A} - \mathbf{C}\}$, inner products $(\mathbf{A} - \mathbf{C})^T\mathbf{W}_0$ and $(\mathbf{F} - \mathbf{G})^T\mathbf{W}_0$ are generated at the upper and lower lag, respectively, of the lower forward–backward tapped-delay-line structure. Note that vectors $\mathbf{F}$, $\mathbf{G}$, and $\mathbf{H}$ are defined in Eqs. (5), (6), and (7), respectively. Inner products of $\mathbf{A}^T(\mathbf{W}_0 + \mathbf{W}_1)$ and $\mathbf{G}^T(\mathbf{W}_0 + \mathbf{W}_1)$ are computed at the top and bottom portions, respectively, of the fast FBLMS architecture. Finally, forward errors $\{e_f(n)$ and $e_f(n-1)\}$ and backward errors $\{e_b(n-1)$ and $e_b(n-2)\}$ are computed at the right-hand side of Fig. 4. In order to subtract the term of $z^{-1}(\mathbf{G} - \mathbf{H})^T\mathbf{W}_1$ and form the backward error, a delay unit is applied to the output branch of the inner product of $\mathbf{G}^T(\mathbf{W}_0 + \mathbf{W}_1)$. Consequently, the corresponding backward error is delayed from $e_b(n)$ to $e_b(n-2)$. Notice that this radix-2 structure concept can be applied again to the upper and lower forward–backward taped-delay-line portion of the fast FBLMS algorithm to further reduce the number of multiplications and additions.

Although the fast FBLMS architecture shown in Fig. 4 appears more complex than the FBLMS shown in Fig. 3, the structure is still very simple. In fact, the fast FBLMS architecture consists of radix-2, forward LMS, and FBLMS structures. The increased data flow complexity over the FBLMS algorithm is limited; therefore, the fast FBLMS algorithm can be easily implemented with digital signal processors.

V. Simulation Results

An adaptive line enhancer with 6-weight ($N = 6$) is chosen as an example. The input signal is a sinusoid of frequency f_0 contaminated by white noise. Computer simulations are conducted for the misadjustment calculation by using forward LMS, FBLMS, and fast FBLMS algorithms. The misadjustment [1] is computed after convergence as follows:

$$\mathbf{M} = \frac{\text{extra output power due to weight jittering}}{\text{minimum output power}}$$

$$= \frac{E[\Delta(n)^T\phi(x,x)\Delta(n)]}{E[e(n)^2]_{opt}} \tag{14}$$

where

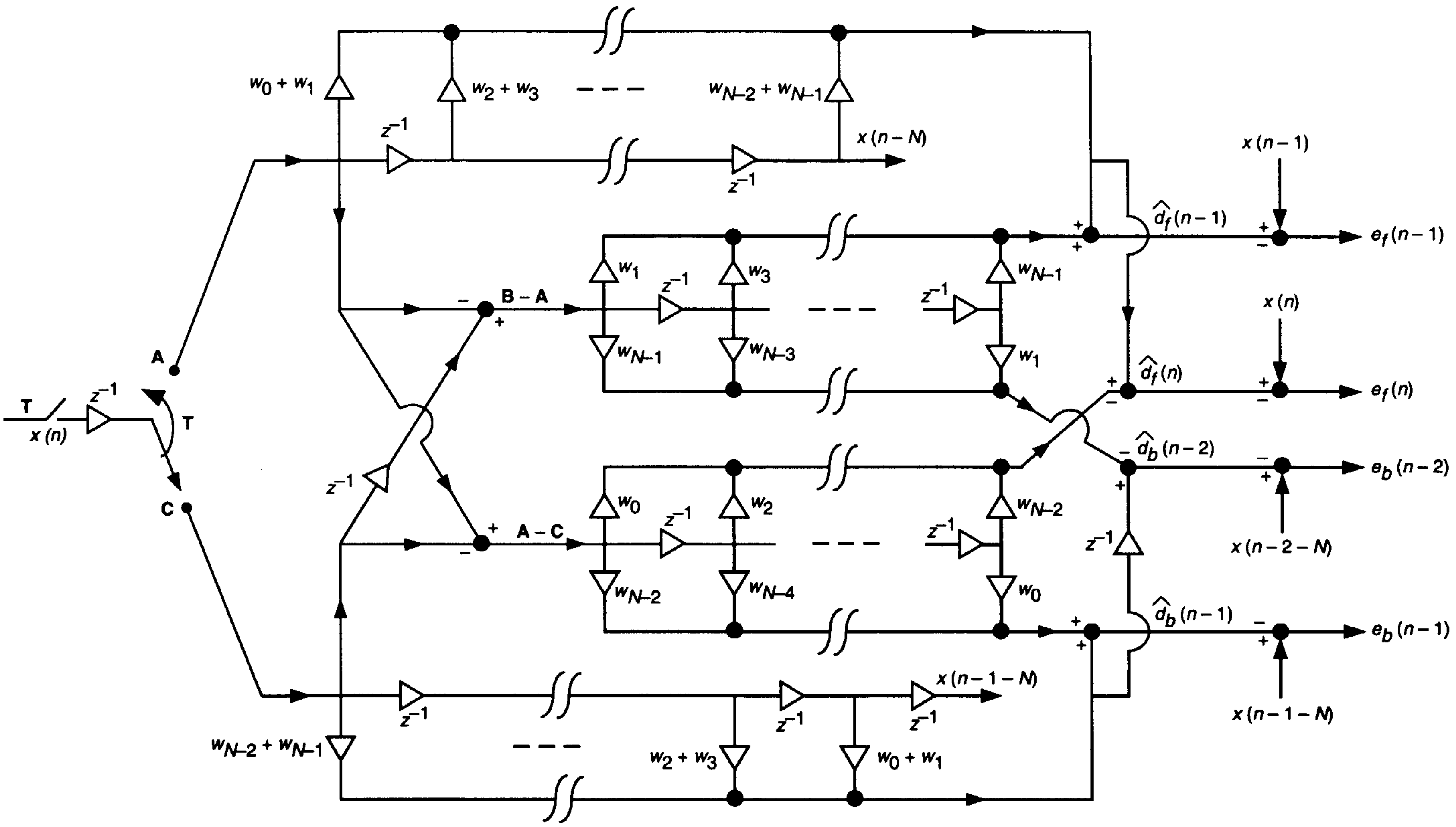

Fig. 4. The architecture of the fast FBLMS algorithm.

$$\Delta(n) = \mathbf{W}(n) - \mathbf{W}_{opt} \tag{15}$$

$$\phi(x, x) = E[\mathbf{X}(n)\mathbf{X}^T(n)] \tag{16}$$

$$E[e(n)^2]_{opt} = E[x(n)^2] - \mathbf{W}_{opt}^T E[x(n)\mathbf{X}(n)] \tag{17}$$

Table 1 shows the measured misadjustments for various values of SNR at step size $\mu = 2^{-8}$. Apparently, the excess error power for both the FBLMS and the fast FBLMS algorithms is approximately half that of the forward LMS algorithm at the 10-dB SNR. The improvement of the misadjustment by using both the FBLMS and the fast FBLMS algorithms over that of the forward LMS algorithm is limited at an SNR around 0 dB. However, the misadjustment of the fast FBLMS algorithm is about the same as that of the FBLMS algorithm. Furthermore, it is observed in Table 1 that, at a higher SNR, the misadjustment increases (for a given step size $\mu = 2^{-8}$). This is because the minimum output error power decreases much more rapidly than the extra output power due to weight jittering, as depicted by Eq. (14). This high misadjustment is significantly reduced when the step size μ is cut to 2^{-10}, as shown in Table 2.

Table 2 shows the measured misadjustments for various values of the step size and the frequency f_0 at SNR = 10 dB. Apparently, the excess error power for both the FBLMS and the fast FBLMS algorithms is approximately half that of the forward LMS algorithm at the step size $\mu = 2^{-8}$ and $\mu = 2^{-10}$. The misadjustment is much reduced when the step size is small (2^{-10}) by using any one of the three algorithms. Again, the misadjustment of the fast FBLMS algorithm is about the same as that of the FBLMS. The $E[e(n)^2]_{opt}$ used to derive the misadjustment is computed by using 500 samples in each run. The misadjustment results listed in Tables 1 and 2 were obtained by averaging 100 runs of the excess error power curves after convergence had been achieved.

Table 1. A comparison between the misadjustment powers of three algorithms at $\mu = 2^{-8}$.

SNR	f_0	Percent misadjustment		
		Forward LMS	FBLMS	Fast FBLMS
0	0.1	3.04	2.75	2.75
3	0.1	3.74	2.84	2.93
10	0.1	32.50	13.77	16.95

Table 2. A comparison between the misadjustment powers of three algorithms using fixed SNR = 10 dB with different μ.

μ	f_0	Percent misadjustment		
		Forward LMS	FBLMS	Fast FBLMS
2^{-8}	0.1667	31.34	14.47	16.03
2^{-8}	0.1	32.5	13.77	16.95
2^{-10}	0.1667	3.06	2.05	1.99
2^{-10}	0.1	2.33	1.24	1.30

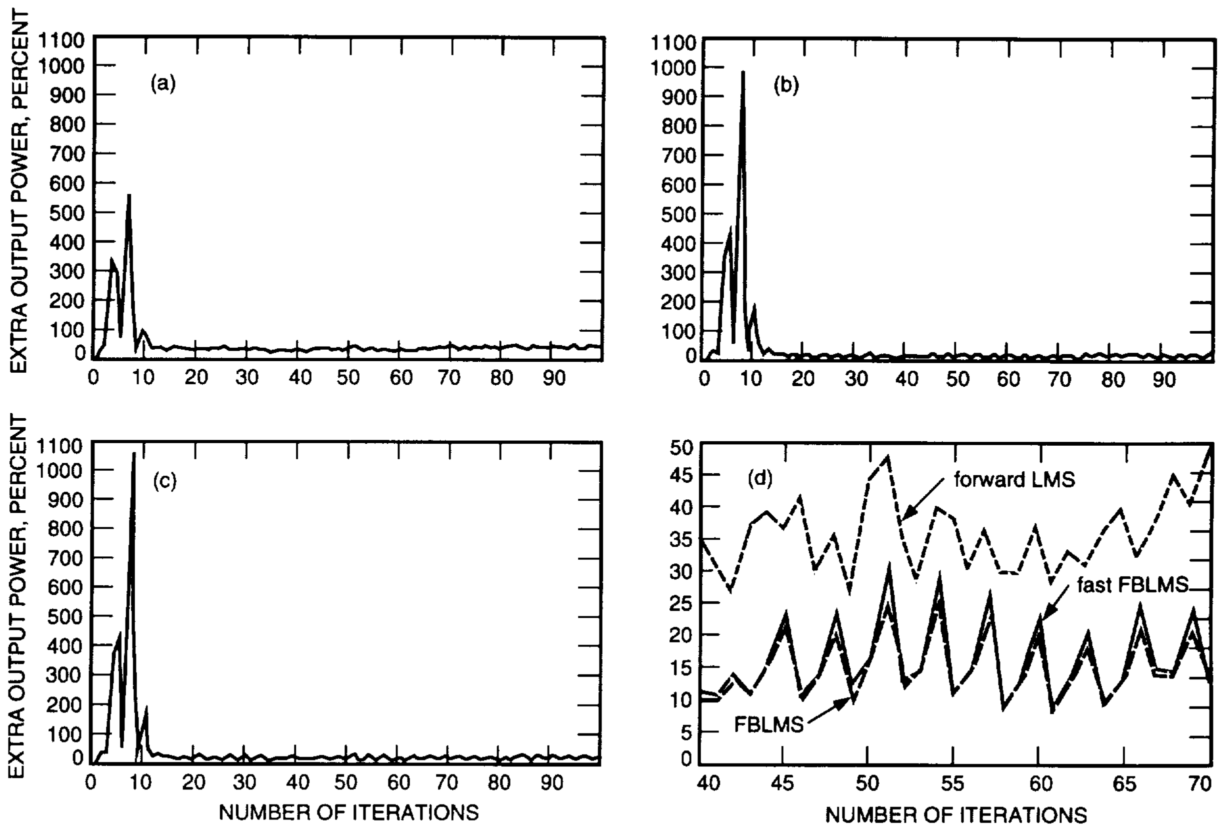

Fig. 5. A typical excess error power versus *n* plot by using the (a) forward LMS, (b) FBLMS, (c) fast FBLMS algorithm, and (d) the steady-state comparison.

Figures 5(a), (b), and (c) show a typical excess error power versus n plot at $f_0 = 1/6$, step size $= 2^{-8}$, and SNR = 10 dB for the forward LMS, FBLMS, and fast FBLMS algorithms, respectively. Figure 5(d) shows the excess error power at the steady state. It is clear that the performance of the fast FBLMS algorithm is about the same as that of the FBLMS algorithm.

VI. Conclusion

The fast forward-backward LMS algorithm presented in this article shows that the number of arithmetic operations in [1] can be reduced without degrading performance. In the forward-backward predictor section, 25 percent of multiplications and 12.5 percent of additions are saved in each of two successive operations. Similarly, in the weight control section, 25 percent of multiplications and 12.5 percent of additions are saved in each of two adaptations. Simulation results indicate that improvements in misadjustment for both the FBLMS and the fast FBLMS algorithms over the conventional LMS algorithm are about 50 percent at a high SNR. When the SNR is low, the misadjustment improvement for both the FBLMS and the fast FBLMS algorithms over the conventional LMS algorithm is less than 50 percent. Notice that this fast forward-backward LMS algorithm is well suited for implementation on application-specific integrated circuits and digital signal processors. This implementation method can be generalized by using higher than two steps of look-ahead. Further computational savings are possible with limited cost on controlling appropriate data flow. This fast FBLMS adaptive line enhancer can be easily integrated together with either a conventional voltage-controlled oscillator in a closed loop for acquisition/tracking, as used in the present deep-space transponder, or a numerically controlled oscillator in an open-loop scheme for acquiring and tracking the carrier signal, as will be used in future deep-space transponders.

Acknowledgments

The authors wish to acknowledge the valuable support of Arthur W. Kermode, Lance A. Riley, and Sami M. Hinedi. Special thanks to Victor A. Vilnrotter for his review, correction, and discussion.

References

[1] Y. C. Lim and C. C. Ko, "Forward-Backward LMS Adaptive Line Enhancer," *IEEE Trans. Circuits Syst.*, vol. 37, no. 7, pp. 936–940, July 1990.

[2] H. G. Yeh and T. M. Nguyen, "Adaptive Line Enhancers for Fast Acquisition," *The Telecommunications and Data Acquisition Progress Report 42-119, July–September 1994*, Jet Propulsion Laboratory, Pasadena, California, pp. 140–159, November 15, 1994.

[3] L. Griffiths, "Rapid Measurement of Digital Instantaneous Frequency," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, no. 2, pp. 207–222, April 1975.

[4] J. T. Rickard, J. R. Zeibler, N. J. Dentino, and M. Shensa, "A Performance Analysis of Adaptive Line Enhancer—Augmented Spectral Detectors," *IEEE Trans. Trans. ASSP*, vol. ASSP-29, pp. 694–701, June 1981.

[5] Z. J. Mou and P. Duhamel, "Short-Length FIR Filters and Their Use in Fast Nonrecursive Filtering," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-39, pp. 1322–1332, June 1991.

[6] K. K. Parhi and D. G. Messerschmitt, "Pipeline Interleaving and Parallelism in Recursive Digital Filters—Part 1: Pipelining Using Scattered Look-Ahead and Decomposition," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-37, pp. 1099–1117, July 1989.