A Trigonometric Formulation of the LMS Algorithm for Realization on Pipelined CORDIC

Mrityunjoy Chakraborty, *Senior Member, IEEE*, A. S. Dhar, and Moon Ho Lee, *Senior Member, IEEE*

***Abstract*—This paper presents an alternate formulation of the least mean square (LMS) algorithm by using a set of angle variables monotonically related to the filter coefficients. The algorithm updates the angles directly instead of the filter coefficients and relies on quantities that can be realized by simple CORDIC rotations. Two architectures based on pipelined CORDIC unit are proposed which achieve efficiency either in time or in area. Further simplifications result from extending the approach to the sign–sign LMS algorithm. An approximate convergence analysis of the proposed algorithm, along with simulation results showing its convergence characteristics are presented.**

***Index Terms*—Least mean square (LMS) algorithm, pipelined CORDIC unit.**

I. Introduction

Since the last two decades, the application of CORDIC arithmetic [1] for efficient implementation of signal processing algorithms continues to receive wide attention because of its numerical stability, efficiency in evaluating trigonometric and hyperbolic functions, hardware compactness and inherent pipelinability at the microlevel [2]. Several algorithms have been reported in the recent past which employ the CORDIC method for performing fast Fourier transform (FFT), discrete cosine transform (DCT), discrete sine transform (DST), singular-value decomposition (SVD), and other matrix operations, filtering and array processing efficiently [2]. For the case of the least-mean-square (LMS)-based adaptive filters, however, the CORDIC approach has so far remained confined largely to lattice filters ([3], [4]) and seemingly has not been extended to the transversal form, as, in the case of the former, the computations in each stage can be related easily to a set of hyperbolic rotations, while no such direct hyperbolic, or, trigonometric interpretation exists for the computations present in the latter. In this paper, we first propose in Section II an alternate formulation of the LMS algorithm by using a set of trigonometric angle variables, which are monotonically related to the transversal filter coefficients. The algorithm updates the angles directly instead of the filter coefficients and relies on quantities that can be realized by simple trigonometric rotations. The resulting architecture is more efficient in terms of hardware and thus power as compared to a direct realization of the adaptive filter, as it replaces about fifty percent of the multipliers by pipelined CORDIC units. Further optimization is also possible by considering the sign–sign version [5] of the proposed algorithm, which eliminates the multipliers completely.

Two architectures are proposed in this paper in Section III based on the presented trigonometric formulation. The first one is a time-efficient architecture which employs a pipelined CORDIC unit and a pipelined multiplier (PM) for each filter tap, resulting in a critical path given by just an adder delay. The second one is an area-efficient architecture, which uses only one pipelined CORDIC unit and a PM, but needs a faster internal clock for multiplexing the filtering as well as weight updating operations for the various taps. This architecture can be viewed as a folded version [7] of the first, though it has been derived purely from intuitive considerations and not by any formal folding transformation.

The proposed trigonometric LMS algorithm has been simulated extensively for evaluation of its convergence behavior. Some of the simulation results are described in Section IV. Exact convergence analysis of the proposed algorithm is, however, an inordinately difficult task due to the presence of sinusoidal nonlinearities in both the filtering and the weight update equations. An approximate convergence analysis and a new upper bound on the step size have been presented in the Appendix.

II. Trigonometric Formulation of LMS Algorithm

We begin by first considering the steepest descent search procedure that arises in the optimal finite-impulse response (FIR) filtering problem. Given an input sequence $x(n)$, desired response $d(n)$ and a N-tap filter coefficient vector $\mathbf{w} = [w_0, w_1, \ldots, w_{N-1}]^t$, the optimal filter $\hat{\mathbf{w}} = [\hat{w}_0, \hat{w}_1, \ldots, \hat{w}_{N-1}]^t$ is obtained by minimizing the mean-square error (MSE) function $\varepsilon^2 = E[e^2(n)]$, where $e(n) = d(n) - \mathbf{w}^t\mathbf{x}(n)$ is the output error, with $\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^t$. The MSE ε^2 is a quadratic function of the filter coefficients w_k's, $k = 0, 1, \ldots, N-1$, and defines the so-called error performance surface in an $N+1$-dimensional space. In the proposed alternative, we first consider an alternate steepest descent search procedure, by selecting a set of N positive numbers A_k's, $k = 0, 1, \ldots, N-1$, so that the minima of the error performance surface is contained in the hypercube with vertices : $[\pm A_0, \pm A_1, \ldots, \pm A_{N-1}]$. (In practice, A_k's are taken to be some powers of 2 for convenience in hardware realization and are chosen using some *a priori* knowledge of the statistics of $x(n)$ and $d(n)$.) Each tap weight w_k in the above range can then be expressed

Manuscript received November 25, 2004; revised February 1, 2005. This work was supported in part by the Institute of Information Technology Assessment (IITA), South Korea. This paper was recommended by Associate Editor S. L. Netto.

M. Chakraborty and A. S. Dhar are with the Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur 721302, India (e-mail: mrityun@ece.iitkgp.ernet.in).

M. H. Lee is with the Department of Information and Communication, Chonbuk National University, Chonju 561 756, South Korea (e-mail: moonho@chonbuk.ac.kr).

Digital Object Identifier 10.1109/TCSII.2005.850784

as $w_k = A_k \sin\theta_k, -(\pi/2) < \theta_k < (\pi/2)$. Since each $w_k \epsilon (-A_k, +A_k)$ maps uniquely to a $\theta_k \epsilon (-(\pi/2), +(\pi/2))$, the MSE ε^2, when expressed as a function of θ_k's has a unique, global minima at $\hat{\theta}_k = \arcsin(\hat{w}_k/A_k), k = 0, 1, \ldots, N-1$, located within a hypercube in the θ space with vertices : $[\pm(\pi/2), \pm(\pi/2), \ldots, \pm(\pi/2)]_{N\times 1}$. Further, the function $\sin\theta_k$ is a monotonically increasing, continuous function of θ_k, as θ_k varies from $-(\pi/2)$ to $+(\pi/2)$, meaning that $\partial\varepsilon^2/\partial\theta_k$ has the same sign as that of $\partial\varepsilon^2/\partial w_k$ everywhere within the hypercube. In other words, the MSE does not exhibit any local minima or maxima within the specified hypercube in the θ space.

In the proposed scheme, a steepest descent search is taken up within the above hypercube in the $\boldsymbol{\theta}$ space in order to reach $\hat{\boldsymbol{\theta}}$. The gradient

$$\nabla_\theta \varepsilon^2 = [\partial\varepsilon^2/\partial\theta_0, \partial\varepsilon^2/\partial\theta_1, \ldots, \partial\varepsilon^2/\partial\theta_{N-1}]^t$$

is easily seen to be given by $\nabla_\theta\varepsilon^2 = -2\boldsymbol{\Delta}(\mathbf{p} - \mathbf{Rw})$, where

$$\begin{aligned}\mathbf{w} &= [A_0 \sin\theta_0, A_1\sin\theta_1, \ldots, A_{N-1}\sin\theta_{N-1}]^t\\ \mathbf{p} &= E[\mathbf{x}(n)d(n)]\\ \mathbf{R} &= E[\mathbf{x}(n)\mathbf{x}^t(n)]\end{aligned}$$

and $\boldsymbol{\Delta}$ is a diagonal matrix with the jth diagonal entry given by $\Delta_{j,j} = A_j\cos\theta_j, j = 0, 1, \ldots, N-1$. The iterate $\boldsymbol{\theta}(i)$ arising in the ith step of iteration is then updated as : $\boldsymbol{\theta}(i+1) = \boldsymbol{\theta}(i) - (\mu/2)\nabla_\theta\varepsilon^2|_{\boldsymbol{\theta}=\boldsymbol{\theta}(i)}$, where μ is an appropriate step size. To move from the steepest descent to the LMS form, we simply replace $\mathbf{R}$ and $\mathbf{p}$ by $\mathbf{x}(n)\mathbf{x}^t(n)$ and $\mathbf{x}(n)d(n)$ respectively in the expression for $\nabla_\theta\varepsilon^2$ in order to obtain an estimate of the gradient at index n. This leads to the so-called "trigonometric LMS (TLMS)" algorithm as follows:

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \mu\boldsymbol{\Delta}(n)\mathbf{x}(n)e(n) \tag{1}$$

$$e(n) = d(n) - \sum_{k=0}^{N-1} A_k \sin\theta_k(n)x(n-k). \tag{2}$$

The TLMS algorithm is particularly suitable for CORDIC-based realization, since the two quantities : $A_k\sin\theta_k(n)x(n-k)$ and $A_k\cos\theta_k(n)x(n-k), k = 0, 1, \ldots, N-1$, required for filtering by and updating of the kth coefficient respectively *can be computed simultaneously by engaging only one CORDIC processor*. Further simplifications are also possible by extending the above treatment to the sign–sign version [5] of the LMS algorithm. In this, each element of the gradient $\boldsymbol{\Delta}(n)\mathbf{x}(n)e(n)$ is replaced by $+1$ or -1, depending on whether it is positive or negative, respectively. Since $\cos\theta_k(n) > 0, -(\pi/2) < \theta_k(n) < +(\pi/2)$, this then leads to the following angle update relation:

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \mu\text{sign}(\mathbf{x}(n))\text{sign}(e(n)). \tag{3}$$

Finally, for pipelined realization, it may, however, be more appropriate to consider the trigonometric analog of an approximate version of the LMS algorithm, popularly known as "delayed LMS" (DLMS) algorithm [6], where the filter coefficients at the nth index are updated using a past estimate of the gradient, say, for index $(n-L)$, where L is an integer. The correction term in the weight update formula then gets modified to $\mu\mathbf{x}(n-L)e(n-L)$ and the resulting L cycle delay in the error feedback path is used for retiming purpose. The trigonometric analog of the DLMS algorithm can be easily worked out by substituting $\mathbf{R}, \mathbf{p}$, and $\mathbf{w}$ in the gradient expression by $\mathbf{x}(n-L)\mathbf{x}^t(n-L), \mathbf{x}(n-L)d(n-L)$ and $[A_0\sin\theta_0(n-L), A_1\sin\theta_1(n-L), \ldots, A_{N-1}\sin\theta_{N-1}(n-L)]^t$, respectively, and is given by

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \mu\boldsymbol{\Delta}(n-L)\mathbf{x}(n-L)e(n-L). \tag{4}$$

The delayed version of the sign–sign angle update formula (3) can similarly be written down by replacing $\mathbf{x}(n)$ and $e(n)$, respectively, by $\mathbf{x}(n-L)$ and $e(n-L)$.

III. Proposed Architectures

We propose two pipelined CORDIC-based architectures, I and II, to implement the delayed TLMS algorithm as given by (2) and (4). Of these, the first one is a time-efficient architecture, while the second one is efficient in area and may be viewed as a folded [7] version of the first. Both architectures employ CORDIC-based rotations to compute $A_k\sin\theta_k(n)x(n-k)$ and $A_k\cos\theta_k(n)x(n-k), k = 0, 1, \ldots, N-1$.

A. CORDIC Algorithm

The CORDIC algorithm first defines a set of elementary angles $\alpha_i, i = 0, 1, 2, \ldots$, where $\alpha_i = \arctan(2^{-i})$. Given any angle θ, with $0° \le \theta < 90°$, it is then possible to express θ as $\theta = \sum_{i=0}^{\infty}\delta_i\alpha_i$, where $\delta_i = \pm 1$, with $\delta_0 = +1$. [For $0° \ge \theta > -90°$, the sign of each δ_i gets reversed.] Rotation of a vector $[X \;\; Y]^t$ by the angle θ then amounts to a sequence of elementary rotations by angles α_i's, forward or backward depending on whether δ_i is positive or negative. In practice, the series is truncated to a finite number of terms, say M, as α_i fast approaches zero as i increases. The CORDIC algorithm can then be summarized as follows.

CORDIC Algorithm: Given the angle $\theta, 0° \le \theta < 90°$ and the vector $[X \;\; Y]^t$, set $x(0) = X, y(0) = Y, \theta(0) = \theta$, and $\delta_0 = +1$. Then

```
For $i = 0$ to $M-1$
Begin
    $\begin{bmatrix} x(i+1) \\ y(i+1) \end{bmatrix} = \begin{bmatrix} 1 & -\delta_i 2^{-i} \\ \delta_i 2^{-i} & 1 \end{bmatrix} \begin{bmatrix} x(i) \\ y(i) \end{bmatrix}$
    $\theta(i+1) = \theta(i) - \delta_i\alpha_i,$
    $\delta_{i+1} = \text{sign}(\theta(i+1)).$
end.
```

Finally, for the sake of correctness, the last iterate $[x(M) \;\; y(M)]^t$ needs to be scaled down by the factor $K_M = \prod_{i=0}^{M-1}\cos\alpha_i$, which, for a given M, is, however, a known constant. In a pipelined CORDIC unit (PCU), the CORDIC rotation is carried out by cascading M identical rotational modules through pipeline latches, with each module responsible for one elementary rotation. The shift implied by 2^{-i} is hardwired and the critical path is given by the time t_a required for a single addition/subtraction.

B. Architecture I

The proposed architecture is shown in Fig. 1 which implements the delayed TLMS algorithm, given by (4). First, the

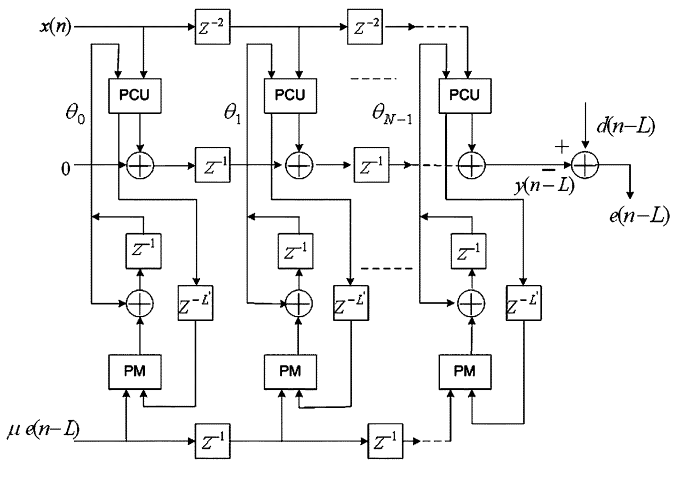

Fig. 1. Time-efficient architecture (Architecture I) to implement the delayed TLMS algorithm; $L' = N - 1$.

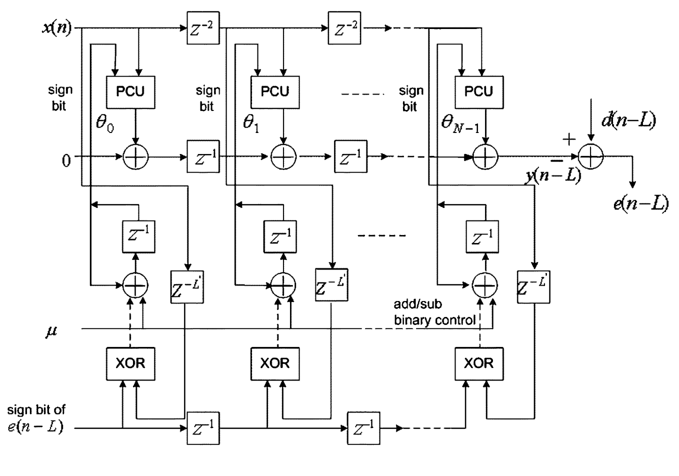

Fig. 2. Time-efficient architecture to implement the sign–sign version of the delayed TLMS algorithm; $L' = N - 1$.

given adaptive filter is pipelined using cutset retiming [7], which introduces a delay each in the paths for $x(n), e(n-L)$ and the one for the intermediate filter output, between every two successive stages. Then, each tap is realized by a PCU and its weight update carried out by a PM as shown. The critical paths of both the PCU and the PM equal one adder delay t_a, meaning that by deploying fast adder structures like the carry save adder [8] or the conditional sum adder [9], an extremely high sampling rate (of the order of hundreds of megahertz) can be employed.

For a total of N number of taps, the architecture has a latency of $L = (N-1)+M$ and an adaptation delay of $(L+M')$, where M and M' are respectively the latency of the PCU and the PM. This implies that the weight updating terms $x\cos\theta$ should be delayed by $L' = N-1$ cycles before being applied to the PM. Finally, the architecture of Fig. 1 gets considerably simplified for the case of the sign–sign version of the delayed TLMS algorithm and is shown in Fig. 2. This replaces each PM by a simple EX-OR gate, which takes the sign bits of the data and the error signals as its input and generates an add/subtract control signal for adding or subtracting μ to $\theta_j(n), j = 0, 1, \dots, N-1$ in the angle update formula (3).

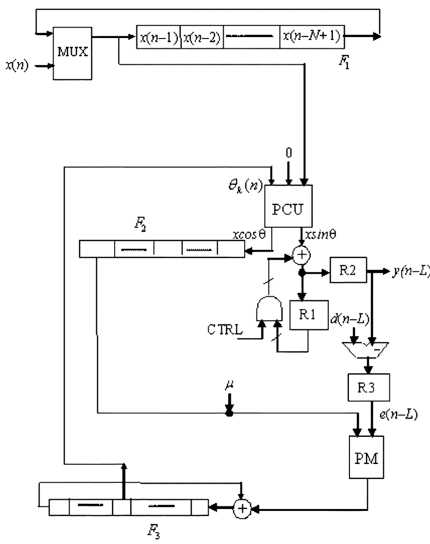

Fig. 3. Area-efficient architecture (Architecture II) to implement the delayed TLMS algorithm.

C. Architecture II

The proposed architecture, as shown in Fig. 3, is an area-efficient architecture which tries to minimize the chip area by trading off the throughput rate. It consists of a PCU, three first-in–first-out (FIFOs) (F_1, F_2, F_3) and a PM, which are clocked by a primary clock N times faster than the input. The maximum frequency of the primary clock is governed by the critical path delay, which in this case also is equal to the propagation delay of a single adder.

The architecture evaluates the filter output $y(n)$ as per the following recursion : $S_0 = 0$ and $S_k = S_{k-1} + x(n-N+k)\sin\theta_{N-k}(n), k = 1, 2, \dots, N$ (we are assuming here A_k's to be 1 for convenience). For this, the partial products are generated by feeding the PCU with the following sequence of data and angles of rotation, in the shown order: $(x(n-N+1), \theta_{N-1}(n)), (x(n-N+2), \theta_{N-2}(n)), \dots, (x(n), \theta_0(n))$. The data samples provided to the PCU are generated by circulating the $N-1$ samples: $x(n-N+k), k = 1, 2, \dots, N-1$ through FIFO F_1 and then flipping the MUX in the Nth cycle to accept the input sample $x(n)$. One of the outputs of the PCU (involving sinusoidal term) is fed to the accumulator built around the register $R1$ that takes N primary clock cycles to complete the computation of $y(n)$. Though the output register $R2$ shares the same input line as the accumulator register $R1$, the former is updated at the sampling frequency f_s, while the latter is clocked at the primary clock frequency of Nf_s. Thus, $R2$ holds the valid output $y(\cdot)$ for the next N primary clock cycles while $R1$ keeps changing in each cycle with partially accumulated product terms. The provision of controlled ANDing facilitates the start of the computation for the next output without losing any clock cycle for clearing the accumulator register $R1$.

It is interesting to note that while one output of the PCU is used for computing $y(n)$, the other output (involving cosinusoidal terms) which is available simultaneously is used for weight updating purpose. For this, a PM, whose number of stages (say, M') depends on the required accuracy, is employed

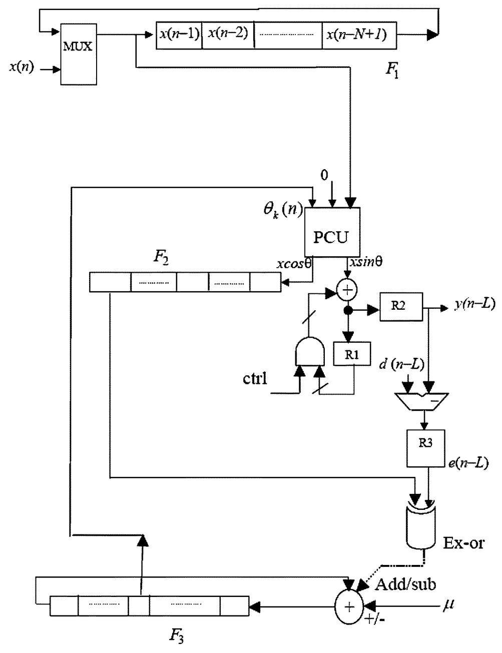

Fig. 4. Area-efficient architecture to implement the sign–sign version of the delayed TLMS algorithm.

to compute the product terms $\mathbf{\Delta}(\cdot)\mathbf{x}(\cdot)e(\cdot)$. To maintain proper synchronization between the indexes of $\mathbf{\Delta}(\cdot)\mathbf{x}(\cdot)$ and $e(\cdot)$, a FIFO (F_2) of length $(N+1)$ is required to be put in the former's data path. In our architecture, the stepsize is assumed to be a power of 2 and thus its multiplication can be implemented by simple hardwired right shift operation without any additional delay. However, in the case of a more general value, its multiplication would take a few clock cycles and the size of the FIFO (F_2) should then be reduced accordingly. Another FIFO (F_3) is employed to store and circulate the angles (θ_k) representing the tap weights. It can be shown easily that the angle required at the input of the PCU for participating in the computation at any particular instant is available at the Kth tap position of F_3 where $K = T$ modulo N, with $T = (M+N+M'+1)$ denoting the total delay from the PCU to the PM. Finally, for the sign–sign case, the PM is replaced by an EX-OR gate which takes the sign bits of $e(n-L)$ and the output of F_2, and generates a control signal that determines whether μ is to be added or subtracted for updating a particular tap weight. The resulting architecture is shown in Fig. 4.

IV. Simulation Studies, Discussion, and Conclusion

In this paper, a trigonometric formulation of the LMS algorithm has been presented which is directly amenable to realization on pipelined CORDIC. An exact convergence analysis of the proposed algorithm is an extremely difficult task owing to the presence of nonlinearities in both the filtering as well as the angle update equations and is not taken up. Instead, an approximate convergence analysis is presented in the Appendix, which provides a new upper bound on the step size for convergence. The convergence behavior of the proposed algorithm is also evaluated by extensive simulation studies. Here, we present simulation results for equalizing an additive white Gaussian noise (AWGN) channel, having a transfer function $(1-2z^{-1})(1-0.5z^{-1})$ and noise variance 0.01. The transmitted symbols were chosen from an alphabet of 5 equispaced, equiprobable discrete amplitude levels with transmitted signal power of 10 dB. A 7-tap equalizer with centre placed at the fourth tap position was used to equalize the channel. First, the TLMS algorithm and its delayed version were employed for training the equalizer with $\mu = 0.0015$ and $\mu = 0.0006$, respectively which generates similar misadjustment for both. The corresponding learning curves, obtained by plotting $E(|e(n)|)$ against n are shown in Fig. 5(a) and (b), respectively. Clearly, the rate of convergence of the delayed TLMS algorithm is less vis-a-vis the TLMS algorithm, which is consistent with the fact that the DLMS algorithm [6] also has slower convergence rate as compared to the LMS algorithm. For the sign–sign TLMS algorithm and its delayed version, however, the convergence rates do not differ much, as seen from the corresponding learning curves shown in Fig. 6(a) and (b), respectively, where,

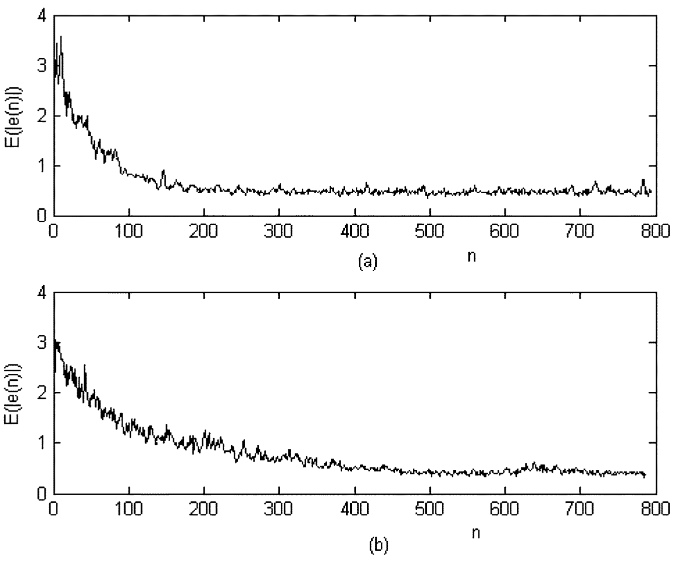

Fig. 5. Learning curves for (a) TLMS algorithm with $\mu = 0.0015$, and (b) delayed TLMS algorithm with $\mu = 0.0006$.

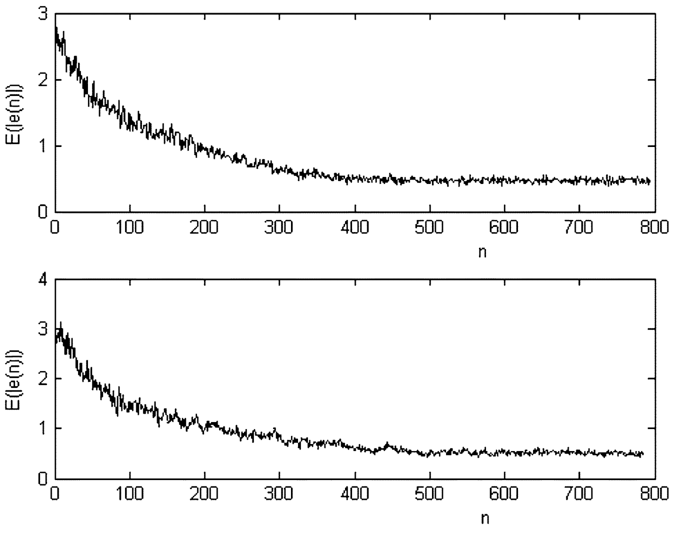

Fig. 6. Learning curves for (a) sign–sign TLMS algorithm with $\mu = 0.007$, and (b) delayed, sign–sign TLMS algorithm with $\mu = 0.005$.

to maintain similar misadjustment for both, the following step size values were chosen: (a) $\mu = 0.007$ (sign–sign TLMS), and (b) $\mu = 0.005$ (delayed sign–sign TLMS).

Appendix
Convergence Analysis

Define the error in the kth angle iterate as $v_k(n) = \theta_k(n) - \hat{\theta}_k, k = 0, 1, \ldots, N-1$. Then, with $\mathbf{v}(n) = [v_0(n), v_1(n), \ldots, v_{N-1}(n)]^t$ and $\hat{\boldsymbol{\theta}} = [\hat{\theta}_0, \hat{\theta}_1, \ldots, \hat{\theta}_{N-1}]^t$, we rewrite (1), after subtracting $\hat{\boldsymbol{\theta}}$ from both sides as

$$\mathbf{v}(n+1) = \mathbf{v}(n) + \mu \boldsymbol{\Delta}(n)\mathbf{x}(n)e(n). \tag{A-1}$$

Next, assuming $v_k(n)$ to be small in magnitude, we make the approximation : $A_k \cos\theta_k(n) = A_k \cos(\hat{\theta}_k + v_k(n)) \approx A_k \cos\hat{\theta}_k - A_k \sin\hat{\theta}_k v_k(n)$, or, equivalently

$$\boldsymbol{\Delta}(n) \approx \hat{\boldsymbol{\Delta}} - \hat{\boldsymbol{\Delta}}'\mathbf{V}(n) \tag{A-2}$$

where, $\hat{\boldsymbol{\Delta}}, \hat{\boldsymbol{\Delta}}'$, and $\mathbf{V}(n)$ are three diagonal matrices with

$$\begin{aligned} \hat{\Delta}_{k,k} &= A_k \cos\hat{\theta}_k, \\ \hat{\Delta}'_{k,k} &= A_k \sin\hat{\theta}_k, \\ [\mathbf{V}(n)]_{k,k} &= v_k(n), \qquad k = 0, 1, \ldots, N-1. \end{aligned}$$

Similarly, $A_k \sin\theta_k(n)$ can be approximated as $A_k \sin\theta_k(n) = A_k \sin(\hat{\theta}_k + v_k(n)) \approx A_k \sin\hat{\theta}_k + A_k \cos\hat{\theta}_k v_k(n) = \hat{w}_k + A_k \cos\hat{\theta}_k v_k(n)$, where $\hat{w}_k = A_k \sin\hat{\theta}_k$ is the kth optimal tap weight. Using this approximation in $\mathbf{w}(n) = [A_0 \sin\theta_0(n), A_1 \sin\theta_1(n), \ldots, A_{N-1} \sin\theta_{N-1}(n)]^t$, we can then write

$$\begin{aligned} e(n) &= d(n) - \mathbf{x}^t(n)\mathbf{w}(n) \\ &= d(n) - \mathbf{x}^t(n)\hat{\mathbf{w}} - \mathbf{x}^t(n)\hat{\boldsymbol{\Delta}}\mathbf{v}(n). \end{aligned} \tag{A-3}$$

Substituting in (A-1), taking expectation of both sides, using the "independence assumption" [11] and recalling that $\mathbf{R}\hat{\mathbf{w}} = \mathbf{p}$, we obtain

$$E[\mathbf{v}(n+1)] = E[\mathbf{v}(n)] - \mu[\hat{\boldsymbol{\Delta}} - \hat{\boldsymbol{\Delta}}' E[\mathbf{V}(n)]]\mathbf{R}\hat{\boldsymbol{\Delta}} E[\mathbf{v}(n)]. \tag{A-4}$$

Neglecting second-order terms involving components of $E[\mathbf{v}(n)]$, or, equivalently, the vector $\hat{\boldsymbol{\Delta}}' E[\mathbf{V}(n)]\mathbf{R}\hat{\boldsymbol{\Delta}} E[\mathbf{v}(n)]$ in comparison to $\hat{\boldsymbol{\Delta}}\mathbf{R}\hat{\boldsymbol{\Delta}} E[\mathbf{v}(n)]$, we get

$$E[\mathbf{v}(n+1)] = [\mathbf{I} - \mu\hat{\boldsymbol{\Delta}}\mathbf{R}\hat{\boldsymbol{\Delta}}]E[\mathbf{v}(n)]. \tag{A-5}$$

Note that the matrix $\hat{\boldsymbol{\Delta}}\mathbf{R}\hat{\boldsymbol{\Delta}}$ is Hermitian and is in fact positive definite if $\mathbf{R}$ is positive definite. From (A-5) and the well-established convergence analysis of the LMS algorithm [11], it is then easy to conclude that $\lim_{n\to\infty} \|E[\mathbf{v}(n)]\| = 0$, if $0 < \mu < (2)/(\mathrm{Trace}[\hat{\boldsymbol{\Delta}}\mathbf{R}\hat{\boldsymbol{\Delta}}])$. For a Toeplitz $\mathbf{R}$ with diagonal entries given by r_0, we have, $\mathrm{Trace}[\hat{\boldsymbol{\Delta}}\mathbf{R}\hat{\boldsymbol{\Delta}}] = r_0 \sum_{k=0}^{N-1} A_k^2 \cos^2\hat{\theta}_k = r_0 \sum_{k=0}^{N-1} A_k^2(1 - \sin^2\hat{\theta}_k) = r_0 \sum_{k=0}^{N-1}(A_k^2 - \hat{w}_k^2)$. This results in the following upper bound of μ for convergence:

$$0 < \mu < \frac{2}{r_0(\|\mathbf{a}\|^2 - \|\hat{\mathbf{w}}\|^2)} \tag{A-6}$$

where $\mathbf{a} = [A_0, A_1, \ldots, A_{N-1}]^t$ and $\hat{\mathbf{w}} = \mathbf{R}^{-1}\mathbf{p}$ is the optimal, i.e., Wiener filter. Note that the closer the vector $\mathbf{a}$ is to $\hat{\mathbf{w}}$, the faster can be the speed of convergence as with increasing upper bound, one can choose larger values for μ.

References

[1] J. E. Volder, "The CORDIC trigonometric computing technique," *IRE Trans. Electron. Comput.*, vol. EC-8, no. 3, pp. 330–334, 1959.

[2] Y. H. Hu, "CORDIC-based VLSI architecture for digital signal processing," *IEEE Signal Process. Mag.*, vol. 9, no. 7, pp. 16–35, Jul. 1992.

[3] Y. H. Hu and H. E. Liao, "CALF : A CORDIC adaptive lattice filter," *IEEE Trans. Signal Process.*, vol. 40, pp. 990–993, 1992.

[4] Y. H. Hu, "On the convergence of the cordic adaptive lattice filtering (CALF) algorithm," *IEEE Trans. Signal Process.*, vol. 46, no. 7, pp. 1861–1871, Jul. 1998.

[5] B. Farhang-Boroujeny, *Adaptive Filters—Theory and Application*. Chichester, U.K.: Wiley, 1998.

[6] G. Long, F. Ling, and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 9, pp. 1397–1405, Sep. 1989.

[7] K. K. Parhi, *VLSI Digital Signal Processing Systems : Design and Implementation*. New York: Wiley, 1999.

[8] A. Weinbergeer, "4:2 Carry-save adder module," *IBM Tech. Discl. Bull.*, vol. 23, no. 8, pp. 3811–3814, Jan. 1981.

[9] J. Sklansky, "Conditional sum addition logic," *IRE Trans. Electron. Comput.*, vol. EC-9, no. 2, pp. 226–230, Jun. 1960.

[10] M. Chakraborty, A. S. Dhar, and S. Pervin, "CORDIC realization of the transversal adaptive filter using a trigonometric LMS algorithm," in *Proc. IEEE ICASSP*, Salt Lake City, UT, May 7–11, 2001, pp. 1225–1228.

[11] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1986.